# Аннотация

Квалификационная работа «Автоматизированная система для запоминания чисел» посвящена разработке веб-приложения, позволяющего быстро переводить числа в соответствующие слова и фразы, сохранять полученный перевод, добавлять слова в базу данных, повышать и понижать рейтинг слов, просматривать статистику использования приложения.

Система состоит из большого числа клиент-серверных компонентов, обеспечивающих работу с пользователем. В системе предусмотрены механизмы отказоустойчивости, которые учитывают возможность одновременного подключения большого числа пользователей.

Расчётно-пояснительная записка включает в себя нескольких частей, в которых подробно описаны этапы и результаты разработки. Конструкторско-технологическая часть содержит обоснование и выбор технологий, модели данных и архитектуры для реализации системы. В научно-исследовательской части представлен выбор алгоритма авторазбиения больших чисел. В экономической части производится расчёт затрат на создание системы.

# Содержание

[Аннотация 3](#_Toc454248460)

[Содержание 4](#_Toc454248461)

[Введение 6](#_Toc454248462)

[1 Конструкторская часть 6](#_Toc454248463)

[1.1 Общетехническое обоснование разработки 6](#_Toc454248464)

[1.1.1 Постановка задачи проектирования 6](#_Toc454248465)

[1.1.2 Описание предметной области 7](#_Toc454248466)

[1.1.3 Перечень процессов, подлежащих автоматизации 8](#_Toc454248467)

[1.1.4 Выбор и обоснование критериев качества 8](#_Toc454248468)

[1.1.5 Анализ аналогов и прототипов 11](#_Toc454248469)

[1.2 Разработка программного изделия 21](#_Toc454248470)

[1.2.1 Архитектура системы 21](#_Toc454248471)

[1.2.2 Инфологическая модель 22](#_Toc454248472)

[1.2.3 Даталогическая модель 27](#_Toc454248473)

[1.2.4 Разработка структуры программного изделия 31](#_Toc454248474)

[1.2.5 Особенности выбранных технологий 32](#_Toc454248475)

[2 Технологическая часть 49](#_Toc454248476)

[2.1 Разработка интерфейса взаимодействия 49](#_Toc454248477)

[2.1.1 Граф диалога 49](#_Toc454248478)

[2.1.2 Интерфейс взаимодействия с пользователем 50](#_Toc454248479)

[2.1.3 Интерфейс взаимодействия с администратором и модератором 60](#_Toc454248480)

[3 Научно-исследовательская часть. Выбор алгоритма авторазбиения чисел 60](#_Toc454248481)

[3.1 Описание предметной области 60](#_Toc454248482)

[3.2 Комбинаторный алгоритм 61](#_Toc454248483)

[3.3 Алгоритм случайного разбиения 61](#_Toc454248484)

[3.4 Выбор наиболее подходящего алгоритма 63](#_Toc454248485)

[4 Организационно-экономическая часть. Расчет затрат на создание программного изделия 71](#_Toc454248486)

[4.1 Смета затрат на программное изделие 71](#_Toc454248487)

[4.2 Обоснование сметы затрат 72](#_Toc454248488)

[4.2.1 Расчёт затрат на расходные материалы 72](#_Toc454248489)

[4.2.2 Расчет затрат на оборудование 72](#_Toc454248490)

[4.2.3 Расчёт затрат на услуги сторонних организаций 73](#_Toc454248491)

[4.2.4 Расчёт заработной платы 74](#_Toc454248492)

[4.2.5 Расходы на дополнительную заработную плату 74](#_Toc454248493)

[4.2.6 Расчет отчислений на социальные нужды 75](#_Toc454248494)

[4.2.7 Расчет накладных расходов 75](#_Toc454248495)

[4.2.8 Расчет прочих расходов 75](#_Toc454248496)

[4.2.9 Расчет себестоимости ИПО 76](#_Toc454248497)

[4.2.10 Расчет цены 76](#_Toc454248498)

[4.2.11 Расчёт чистой прибыли до и после налогообложения 76](#_Toc454248499)

[Заключение 76](#_Toc454248500)

[Список использованных источников 78](#_Toc454248501)

# Введение

На сегодняшний день люди предпочитают хранить числовую информацию (логины, пароли, номера телефонов, пин-коды), записывая в блокноте или в файл на компьютере. Информация, хранящаяся в таком виде, легко взламывается. Поэтому разработка приложения, которое предоставит почву для альтернативного варианта хранения числовой информации, является актуальной. В том числе, разработка такого приложения позволит людям чаще тренировать свою ассоциативную память и абстрактное мышление.

# Конструкторская часть

## Общетехническое обоснование разработки

### Постановка задачи проектирования

Целью данной работы является проектирование и реализация автоматизированной системы, состоящей из серверного и клиентского программного обеспечения. В системе предусматриваются механизмы отработки отказа в случае выхода из строя одного или нескольких принимающих данные серверов, а также при необходимости осуществления замены и ремонта серверного оборудования. Также нужно предусмотреть возможность масштабирования системы.

В системе предусмотрены автоматизированные рабочие места *операторов*, с которых можно осуществлять работу с системой: искать и добавлять слова, посещать страницы других пользователей, влиять на рейтинг слов и фраз.

Исходя из вышенаписанного, задачи проектирования могут быть сформулированы следующим образом:

* исследование и анализ предметной области;
* поиск и выбор аналогов;
* разработка архитектуры системы;
* разработка интерфейсов взаимодействия;
* разработка модели базы данных;
* написание и отладка программного кода;
* разработка технической документации.

### Описание предметной области

Большинство людей по всему миру предпочитают не полагаться полностью на свою память, чтобы запомнить важную информацию. Числовая информация (пин-коды, пароли, логины, номера телефонов и банковских карточек), которую люди сейчас предпочитают на всякий случай сохранять в незашифрованном, а записанном в блокноте или файле компьютера виде, легко взламывается злоумышленниками и мошенниками. [25]

Числа не провоцируют появление в мозгу образов, поэтому на данное время популярным методом для запоминания чисел являются мнемонические правила. Мнемонические правила также являются отличной тренировкой для ассоциативной памяти и абстрактного мышления человека. [22]

Отсюда следует необходимость в автоматизированной системе для запоминания чисел, которая просто и быстро, с помощью мнемонических правил, переводила бы числа в слова и фразы. Последние в свою очередь провоцировали бы появление в мозгу человека запоминаемых образов, благодаря которым необходимость хранения числовой информации на бумажном или файловом носителе отпадает. Также такой образ может найти свое воплощение в нарисованной картинке, точно название (или значение) которой будет известно только нарисовавшему. Следовательно, узнать пароль или пин-код по такому изображению злоумышленнику гораздо сложнее.

Система должна быть простой для понимания. Также необходимо сделать ее простой для освоения пользователем: он должен быстро понять, как шифруются и расшифровываются числа. Для этого в системе будет автоматизирован цифро-буквенный метод. Его подробное описание можно найти в пункте 1.1.5.1.

### Перечень процессов, подлежащих автоматизации

1. добавление, редактирование, удаление слов;
2. цифро-буквенный перевод числа в слово и наоборот;
3. поиск слов и фраз, соответствующих числовым комбинациям;
4. повышение (понижение) рейтинга слов и фраз при поиске;
5. добавление, редактирование, удаление пользователей;
6. сохранение (удаление) запросов пользователя;
7. повышение (понижение) рейтинга страницы пользователя посредством повышения (понижения) рейтинга его сохраненных запросов;
8. сбор статистики использования системы,
9. разбиение больших чисел на группы цифр.

### Выбор и обоснование критериев качества

При проектировании системы для запоминания чисел важными являются следующие критерии качества:

1. время освоения;
2. скорость работы;
3. удобство работы;
4. автоматизируемость;
5. применимость.

#### Время освоения

Пользователь должен быстро разобраться и освоить систему. Время освоения не должно превышать 24 часов.

#### Скорость работы

Скорость работы включает требования к временным характеристикам:

* время формирования запросов и выдачи результатов не должно превышать 30-60 секунд;
* время работы программы не должно превышать 30-60 секунд (зависит от скорости передачи информации в локальной или глобальной сети).

#### Удобство работы

Удобство работы означает понятность системы и простоту работы с ней. Обеспечение удобства интерфейса позволяет минимизировать время, затрачиваемое при работе с системой и получать максимальную эффективность работы. Интерфейс, основанный на web-страницах, является наиболее доступным и удобным, так как необходимо только наличие браузера, а необходимость в установке программного продукта отсутствует.

#### Автоматизируемость

Автоматизируемость означает наличие программных реализаций у системы.

#### Применимость

Применимость означает возможность применения алгоритмов, используемых в системе, к другим предметным областям и задачам.

#### Оценка критериев качества

Вышеупомянутые критерии были сведены в таблицы для последующего принятия решения, используя в качестве интегрального критерия взвешенную сумму показателей сравнения. В табл. 1 приведены обозначения критериев и их краткое описание. В табл. 2 приведена вербально числовая шкала (критерии качества, словесному и числовому описанию которых приведена числовая оценка предпочтительности).

Таблица 1 Критерии для сравнения аналогов

|  |  |  |
| --- | --- | --- |
| Критерий | Обозначение | Описание критерия |
| Время освоения |  | Время, за которое пользователь разбирается с программой и начинает уверенное пользование |
| Скорость работы |  | Время выдачи результатов и работы программы |
| Удобство работы |  | Понятность и простота работы системы |
| Автоматизируемость |  | Наличие программных реализаций |
| Применимость |  | Возможность использования алгоритмов в других предметных областях |

Таблица 2 Вербально-числовая шкала оценки критериев вариантов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Оценка |  |  |  |  |  |
| 5 | <5мин | <1c | Интуитивно-понятный интерфейс, действия выполняются быстро и просто | Полностью автоматизируема | Алгоритмы применимы к большинству любых предметных областей |
| 4 | 5-14мин | 1-2с | Понятный интерфейс, но для выполнения действий нужно немного времени | Автоматизирована большая часть | Алгоритмы применимы к большинству похожих предметных областей |
| 3 | 15-29мин | 3-5с | Понятный, но местами запутанный интерфейс. Для выполнения действий нужно немного времени | Частично автоматизирована | Алгоритмы применимы лишь к нескольким сходным предметным областям |
| 2 | 30-60часа | 6-60с | Запутанный интерфейс. Для выполнения действий нужно немного времени | Автоматизация нуждается в существенных доработках | Одни алгоритмы системы применимы к другим предметным областям, а другие применимы только в одной конкретной предметной области |
| 1 | >1часа | >60c | Запутанный интерфейс. Для выполнения действий требуется много времени. | Неавтоматизируема | Применим только в одной конкретной предметной области |

### Анализ аналогов и прототипов

Заслуживающими внимание альтернативами являются:

1. Ручной труд – перевод числа в соответствующие слова и фразы посредством Excel или на бумаге;
2. ЧДП – техника Человек-Действие-Предмет, где каждому числу присваивается образ вида человек делает что-то с чем-то.

#### Ручной труд

Ручной труд представляет собой подбор слов и фраз к числа посредством вычислений в Excel или на бумаге.

Существует несколько видов ручного подбора слов:

1. Использование звуковой памяти. В одном варианте этого способа подбирается фраза, в которой число букв в каждом слове соответствует очередной цифре числа. Например:

«Это я знаю и помню прекрасно,

Пи многие знаки мне лишни, напрасны»

Соответственно расшифровывается как: 3,14159265358.

Во втором варианте рифмуются сами цифры. Например:

«Чтобы ПИ запомнить, братцы,

Надо чаще повторять:

Три, четырнадцать, пятнадцать,

Девять, двадцать шесть и пять»

Используя этот метод, легко подбирать слова и составлять из них предложения, а так же расшифровывать, но в случае, например, 7-значного номера телефона, может получиться очень длинное предложение. А если составлять длинные предложения даже в небольшом количестве, можно запутаться.

2. Цифро-буквенный алфавит

Суть этого метода заключается в том, что каждой цифре ставятся в соответствие какие-то буквы, а из букв составляются слова. Известно несколько таких алфавитов. Вот один из них:

0. Буквы Н и Л в слове «ноль».

1. Р от слова «раз».

2. Д — «Два», так как эта цифра больше всего напоминает данную маленькую букву.

3. Т от «Три» и 3 — по внешнему сходству.

4. Ч от слова «Четыре» из-за внешнего сходства и звучания.

5. П — «Пять»по схожести звучания.

6. Ш — «Шесть» - тоже по схожести звучания.

7. С — «Семь».

8. В — «Восемь».

9. Здесь сложнее. Можно М от слова «Много», так как 9 самая большая цифра, и З из-за внешнего сходства.

Например, число 3312 можно записать как «тетрадь».

3. Использование раскладки телефонной клавиатуры

Цифры заменяются соответствующими буквами из телефонной клавиатуры, а из букв составляются слова. Например: Телефонный номер Яндекса: 8-800-333-96-39, для запоминания предложен текстовый вариант номера: 8-800-333-YNDX.

Цифровой ряд запоминается легче, если заменить сложные для восприятия его части буквами, образующими некую фразу, либо слово. Недостатком метода является то, что цифры 0 и 1 не имеют буквенных значений в телефонной клавиатуре, поэтому запоминать их с помощью этого способа не получится, что является большим недостатком, так как это 1/5 всех существующих цифр (речь идет о 10-тичной системе счисления, в которой считает большинство людей).

При использовании этого способа получаются маленькие фразы, так как в одном слове может быть закодировано несколько цифр. Также числа легко расшифровывать, но это зависит от кодировки: есть интуитивно-понятные шифры (как приведенный выше), а есть те, что нужно заучивать как таблицу умножения.

Однако необходимо подбирать слова по получившимся буквам и предложения по словам, а это, иногда, бывает затруднительным. А так же запоминать «кодировку» — соответствие цифрам определенных букв. Для упрощения этого процесса можно вести тетрадь и автоматизировать подбор в Excel. [25]

#### ЧДП

Эта техника имеет прямое отношение к мнемонике и более того, именно ей пользуются на различных чемпионатах по запоминанию.

ЧДП расшифровывается как Человек-Действие-Предмет. Ее суть состоит в том, чтобы каждому числу от 1 до 99 присваивается свой образ вида какой-то человек делает что-то с чем-то. Например, число 10 в ЧДП — это мышь скребет стену, 11 — Эйнштейн показывает язык, а 12 — йог медитирует на столе.

Теперь, когда будет необходимо запомнить последовательность цифр 101211, нужно будет делать это следующим образом: 10 — животное — мышь, 12 — действие — медитирует, 11 — предмет — язык. Таким образом, последовательность цифр 101211 кодируется в один образ: мышь медитирует на языке. И тут можно дать просто для фантазии: мышь медитирует на летающем языке, на горе из языков и т.д.

Для того чтобы образы были запоминающимися, они должны быть цветными, объемными, подвижными и, разумеется, необычными. Таким образом, хоть система ЧДП и предполагает множество незабываемых образов (порядка миллиона уникальных образов), нужно все равно стараться придумать что-то максимально оригинальное. [22]

Существует еще одна вариация этого способа. Она заключается в следующем. Цифру сопоставляют ассоциируемому слову. Например так:

1) Одна цифра:

Единица — кол, деревяшка в ошмётках коры, шершавая на ощупь. Двойка — лебедь, белоснежный и толстый и так далее для всех цифр.

2)Две и более:

После создания образного ряда запоминание цифр превращается в игру, похожую на рисование мультфильма.

Например, для запоминания числа 21 можно представить себе лебедя, на которого упал кол, ушиб его и лебедь злобно шипит и щиплет кол. Таким образом, вместо абстрактных цифр в памяти появляются движущиеся картинки, которые легко и прочно в ней удерживаются.

Фантазия человека всегда найдет образное сравнение для любой комбинации цифр, однако при запоминании, к примеру, телефона получается длинный «мультик», а если телефонов несколько, то картинки из «мультиков» могут перепутаться между собой. [25]

#### Анализ методом взвешенной суммы

Сравним разрабатываемую в рамках выпускной работы систему с двумя аналогами.

Необходимо произвести анализ не только среди вариантов систем по различным критериям, но и учесть относительную важность критериев.

Таблица 3 Варианты аналогов систем для запоминания чисел

|  |  |
| --- | --- |
| Вариант | Описание |
|  | Ручной труд – перевод числа в соответствующие слова и фразы посредством Excel или на бумаге. |
|  | ЧДП – техника Человек-Действие-Предмет, где каждому числу присваивается образ вида человек делает что-то с чем-то. |
|  | Автоматизированная система для запоминания чисел, которой посвящена данная выпускная работа |

В табл. 3 приведены варианты аналогов систем для запоминания чисел.

В табл. 4 приведены значения критериев для первого варианта.

Таблица 4 Значения критериев для первого варианта

|  |  |
| --- | --- |
| Критерий | Значение |
|  | 20 минут |
|  | В среднем 4-5 секунд |
|  | Понятный интерфейс, но для выполнения действий требуется немного времени |
|  | Автоматизация нуждается в существенных доработках |
|  | Алгоритм применим лишь в нескольких сходных предметных областях |

В табл. 5 приведены значения критериев для второго варианта.

Таблица 5 Значения критериев для второго варианта

|  |  |
| --- | --- |
| Критерий | Значение |
|  | 10 минут |
|  | В среднем 6-7 секунд |
|  | Понятный интерфейс, но для выполнения действий требуется немного времени |
|  | Не существует программных реализаций, неавтоматизируема |
|  | Алгоритм применим только в одной конкретной области |

В табл. 6 приведены значения критериев для третьего варианта.

Таблица 6 Значения критериев для третьего варианта

|  |  |
| --- | --- |
| Критерий | Значение |
|  | 6-10 минут |
|  | <1с |
|  | Интуитивно-понятный интерфейс |
|  | Полностью автоматизируема |
|  | Алгоритм применим лишь в нескольких сходных предметных областях |

Для исключения неэффективных альтернативных вариантов, т. е. вариантов, которые не лучше других рассматриваемых вариантов по всем локальным критериям, будем использовать метод отбора парето-оптимальных вариантов. Этот подход позволяет уменьшить количество вариантов, подлежащих дальнейшему сравнению. Будем использовать следующий алгоритму:

1) построим таблицу исходных вариантов, подлежащих сравнению. В этой таблице указывают — численное значение -го варианта по -му критерию;

2) построим Парето-оптимальную квадратную таблицу А размером , где — число сравниваемых вариантов. Присвоим диагональным элементам таблицы значения, равные нулю, а элементам — значения , если вариант предпочтительнее варианта по всем критериям, и , если вариант уступает или не лучше варианта хотя бы по одному критерию. Если в -м столбце таблицы А все элементы равны нулю, то соответствующие варианты относятся к Парето-оптимальным, а если присутствует хотя бы одна единица, то такие варианты исключаются из дальнейшего рассмотрения как неэффективные. [20]

Таблица 7 Численные значения критериев вариантов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерий  Вариант |  |  |  |  |
|  | 3 | 4 | 4 | 0,26 |
|  | 3 | 2 | 5 | 0,13 |
|  | 4 | 4 | 5 | 0,26 |
|  | 2 | 1 | 5 | 0,26 |
|  | 3 | 1 | 3 | 0,067 |

Таблица 8 Сравнение вариантов систем на Парето-оптимальность

|  |  |  |  |
| --- | --- | --- | --- |
| Варианты |  |  |  |
|  | 0 | 0 | 0 |
|  | 0 | 0 | 0 |
|  | 0 | 0 | 0 |
| Парето-оптимальность варианта | Да | Да | Да |

Анализ данных, приведенных в таблицах 7 и 8, показывает, что все варианты сравниваемых систем являются Парето-оптимальными и требуется проведение их дальнейшего сравнения.

Для вычисления коэффициентов важности локальных критериев применим метод базового критерия. Будем придерживаться следующего порядка действий:

1) разобьем все множество локальных критериев на группы важности, при этом самые малозначимые критерии относим к первой (базовой) группе.

2) составим уравнение нормировки локальных критериев, в котором сумма коэффициентов важности всех локальных критериев должна быть равна единице.

. (1.1)

Где - количество групп важности локальных критериев;

- количество локальных критериев, которые входят в состав -ой группы

- коэффициент, который показывает степень превосходства критериев -ой группы по сравнению с критериями базовой группы, при этом

- коэффициент важности критериев первой группы, при

3) определим коэффициент важности () критериев базовой группы

(1.2)

4) определим коэффициенты важности () локальных критериев, входящих в состав -ой группы

(1.3)

5) выполним проверку условия нормировки локальных критериев

(1.4)

Следует иметь в виду, что

(1.5)

Где - количество локальных критериев, по которым проводим сравнение альтернативных вариантов систем.

Наиболее часто используют интегральный критерий взвешенной суммы для выбора наилучшего варианта из набора альтернативных вариантов систем. При этом локальными критериями являются показатели, по которым производят сравнение.

(1.6)

где — число показателей сравнения; — количество вариантов сравнения; cравнение вариантов систем ( ) осуществляется по параметрам ();

— коэффициент важности -го параметра сравнения. При этом обязательно должно быть выполнено условие нормировки критериев;

— коэффициент нормализации, определяет уровень соответствия -го параметра -го варианта наилучшему значению, .

Наилучшим вариантом системы является вариант .

При нормализации критериев типа «чем больше, тем лучше» (например, производительность, гарантийный срок работы и т. д.) коэффициент нормализации находят согласно выражению

(1.7)

При нормализации критериев типа «чем больше, тем хуже» (например, габариты, масса, стоимость и т. д.) коэффициент нормализации определяют из выражения

(1.8)

где — значение -го локального критерия, соответствующее максимальному значению среди сравниваемых вариантов решения;

— значение -го локального критерия, соответствующее минимальному значению среди сравниваемых вариантов решения. [20]

Разобьем все показатели систем на группы важности.

В первую группу включаем один показатель (с кодовым обозначением , который считаем самым малозначимым из набора показателей.

Во вторую группу включаем один показатель (с кодовым обозначением , который считаем более значимым, по сравнению с первыми в два раза.

В третью группу включаем три показателя (с кодовыми обозначениями ), которые считаем более значимыми по сравнению с первыми, в 4 раза.

Поэтому имеем следующие исходные данные:

- количество групп показателей сравнения систем;

- количество показателей, которые соответственно входят в состав 1-ой, 2-ой и 3-ей групп.

- коэффициенты, которые соответственно показывают степень превосходства критериев 2-ой и 3-ей группы над критериями 1-ой группы.

После подстановки данных в выражение 1.1 имеем:

Решая это уравнение, получаем

Подставляем вычисленное выражение ( в выражение 1.3 и получаем значения коэффициентов важности показателей сравнения систем:

Рассчитанные значения коэффициентов важности показателей сравнения систем, а также нормализованные значения этих показателей, вычисленные по формулам 1.7 и 1.8, приведены в таблице 9.

Таблица 9 Значения интегрального критерия

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код локального критерия | Коэффициент важности локального критерия ( | Нормированное значение локального критерия | | |
|  |  |  |
|  | 0,26 | 0,75 | 1 | 1 |
|  | 0,13 | 0,6 | 0,4 | 1 |
|  | 0,26 | 0,8 | 0,8 | 1 |
|  | 0,26 | 0,4 | 0,2 | 1 |
|  | 0,067 | 1 | 0,3 | 1 |
|  | | 0,652 | 0,5921 | 1 |

Значения интегрального критерия приведены в таблице 7 (в последней строке) и представляют собой аддитивную свертку локальных критериев. Ранжирование вариантов систем показывает, что системы следует расположить в следующем порядке .При этом вариант системы с кодовым названием является наилучшим среди сравниваемых.

В итоге получили, что разрабатываемая система является лучшей среди аналогов и прототипов.

## Разработка программного изделия

### Архитектура системы

Архитектура автоматизированной системы для запоминания чисел приведена на рис. .

|  |
| --- |
| ArchSystem.png  Рис. 1 Архитектура системы |

### Инфологическая модель

Инфологическая модель строится по результатам анализа предметной области, в итоге выделяются объекты и связи между ними. [24]

#### Сущности и их атрибуты

Таблица 10 Сущности и их атрибуты

|  |  |  |  |
| --- | --- | --- | --- |
| Логическое имя атрибута | Перевод | Тип данных | Ключ |
| Сущность «Слово» | | | |
| Номер\_слова | wordId | Числовой | Первичный |
| Слово | word | Текстовый |  |
| Числовой\_базовый\_ эквивалент | num\_base | Текстовый |  |
| Числовой\_дополнительный\_ эквивалент | num\_dop | Текстовый |  |
| Числовой\_ смешанный\_эквивалент | num\_mix | Текстовый |  |
| Вес | weight | Числовой |  |
| Сущность «Связь между словами» | | | |
| Номер\_связи | linkId | Числовой | Первичный |
| Вес\_связи | weight | Числовой |  |
| Номер\_слова1 | wordId | Числовой | Внешний |
| Номер\_слова2 | wordId | Числовой | Внешний |
| Сущность «Пользователь» | | | |
| Номер\_пользователя | userId | Числовой | Первичный |
| Дата\_регистрации | date | Дата |  |
| Суперпользователь | superuser | Булевый |  |
| Номер\_ВКонтакте | vkid | Числовой |  |
| Имя | username | Текстовый |  |
| Сущность «Статистика» | | | |
| Номер\_статистики | statId | Числовой | Первичный |
| Дата | date | Дата |  |
| Ключ | key | Текстовый |  |
| Счетчик | counter | Числовой |  |
| Cущность «Плохое слово» | | | |
| Номер\_плохого\_слова | ProfanityId | Числовой | Первичный |
| Слово | word | Текстовый |  |
| Сущность «Понравившийся запрос» | | | |
| Номер\_понравившегося\_запроса | phraseId | Числовой | Первичный |
| Номер\_отметившего | userId | Числовой | Внешний |
| Номер\_содержащего | userId | Числовой | Внешний |
| Номер\_запроса | savedqueryId | Числовой | Внешний |
| Сущность «Сохраненный запрос» | | | |
| Номер\_сохраненного\_запроса | savedqueryId | Числовой | Первичный |
| Число | num | Текстовый |  |
| Фраза | phrase | Текстовый |  |
| Вес | weight | Числовой |  |
| Номер\_сохранившего | userId | Числовой | Внешний |
| Сущность «Добавленное слово» | | | |
| Номер\_слова | wordId | Числовой | Первичный |
| Слово | word | Текстовый |  |
| Числовой\_базовый\_эквивалент | num\_base | Текстовый |  |
| Числовой\_дополнительный\_эквивалент | num\_dop | Текстовый |  |
| Числовой смешанный эквивалент | num\_mix | Текстовый |  |
| Вес | weight | Числовой |  |
| Дата | date | Дата |  |
| Номер\_добавившего | userId | Числовй | Внешний |

#### Связи между сущностями

Таблица 11. Таблица связей между сущностями

|  |  |  |  |
| --- | --- | --- | --- |
| Название связи | Первая сущность | Вторая сущность | Тип связи |
| Связывается\_слева | Слово | Связь между словами | 1:М |
| Связывается\_справа | Слово | Связь между словами | 1:М |
| Добавил | Добавленное слово | Пользователь | M:1 |
| Сохранил | Пользователь | Сохраненный  запрос | 1:М |
| Отметил | Пользователь | Понравившийся запрос | 1:М |
| Содержит | Пользователь | Понравившийся запрос | 1:М |
| Понравился | Сохраненный запрос | Понравившийся запрос | 1:М |

#### Схема инфологической модели

Схема инфологической модели базы данных системы в нотации Питера Чена приведена на рис. .

|  |
| --- |
| инфо.png  Рис. 2 Инфологическая модель предметной области |

### Даталогическая модель

Даталогическая модель строится на основе инфологической модели. В даталогической модели сущности соответствуют таблицам, в то время как атрибуты соответствуют полям таблиц. [24]

Таблица 12 Таблица Word

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Логическое имя атрибута | Поле | Тип данных | Длина поля в байтах | NN (Not Null) | Тип ключа |
| Номер\_слова | wordId | INTEGER | 4 | True | PK |
| Слово | word | VARCHAR | 25 | True |  |
| Числовой\_базовый\_ эквивалент | num\_base | VARCHAR | 25 | True |  |
| Числовой\_дополнительный\_ эквивалент | num\_dop | VARCHAR | 25 | True |  |
| Числовой\_ смешанный\_эквивалент | num\_mix | VARCHAR | 25 | True |  |
| Вес | weight | INTEGER | 4 | True |  |

Таблица 13 Таблица WordsLinksWeight

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Логическое имя атрибута | Поле | Тип данных | Длина поля в байтах | NN (Not Null) | Тип поля | Тип ключа |
| Номер\_связи | linkId | INTEGER | 4 | True |  | PK |
| Вес\_связи | weight | INTEGER | 4 | True | Unique |  |
| Номер\_слова1 | wordId | INTEGER | 4 | True | Unique | FK |
| Номер\_слова2 | wordId | INTEGER | 4 | True |  | FK |

Таблица 14 Таблица User

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Логическое имя атрибута | Поле | Тип данных | Длина поля | NN (Not Null) | Тип поля | Тип ключа |
| Номер\_пользователя | userId | INTEGER | 4 | True |  | PK |
| Дата\_регистрации | date | DATE | 4 | True |  |  |
| Суперпользователь | superuser | BOOLEAN | 1 | True |  |  |
| Номер\_ВКонтакте | vkid | INTEGER | 4 | True | Unique |  |
| Имя | username | VARCHAR | 50 | True |  |  |

Таблица 15 Таблица Statistics

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Логическое имя атрибута | Поле | Тип данных | Длина поля | NN (Not Null) | Тип поля | Тип ключа |
| Номер\_статистики | statId | INTEGER | 4 | True |  | PK |
| Дата | date | DATE | 4 | True | Unique |  |
| Ключ | key | VARCHAR | 25 | True | Unique |  |
| Счетчик | counter | INTEGER | 4 | True |  |  |

Таблица 16 Таблица Profanity

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Логическое имя атрибута | Поле | Тип данных | Длина поля | NN (Not Null) | Тип ключа |
| Номер\_плохого\_слова | ProfanityId | INTEGER | 4 | True | PK |
| Слово | word | VARCHAR | 25 | True |  |

Таблица 17 Таблица LikedQuery

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Логическое имя атрибута | Поле | Тип данных | Длина поля | NN (Not Null) | Тип поля | Тип ключа |
| Номер\_понравившегося\_запроса | phraseId | INTEGER | 4 | True |  | PK |
| Номер\_отметившего | userId | INTEGER | 4 | True | Unique | FK |
| Номер\_содержащего | userId | INTEGER | 4 | True |  | FK |
| Номер\_запроса | savedqueryId | INTEGER | 4 | True | Unique | FK |

Таблица 18 Таблица SavedQuery

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Логическое имя атрибута | Поле | Тип данных | Длина поля | NN (Not Null) | Тип поля | Тип ключа |
| Номер\_сохраненного\_запроса | savedqueryId | INTEGER | 4 | True |  | PK |
| Число | num | INTEGER | 4 | True | Unique |  |
| Фраза | phrase | VARCHAR | 25 | True | Unique |  |
| Вес | weight | VARCHAR | 25 | True | Unique |  |
| Номер\_сохранившего | userId | INTEGER | 4 | True |  | FK |

Таблица 19 Таблица AddedWord

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Логическое имя атрибута | Поле | Тип данных | Длина поля | NN (Not Null) | Тип ключа |
| Номер\_слова | wordId | INTEGER | 4 | True | PK |
| Слово | word | VARCHAR | 25 | True |  |
| Числовой\_базовый\_эквивалент | num\_base | VARCHAR | 25 | True |  |
| Числовой\_дополнительный\_эквивалент | num\_dop | VARCHAR | 25 | True |  |
| Числовой смешанный эквивалент | num\_mix | VARCHAR | 25 | True |  |
| Вес | weight | INTEGER | 4 | True |  |
| Дата | date | INTEGER | 4 | True |  |
| Номер\_добавившего | userId | DATE | 4 | True | FK |

Исходя из построенной даталогической модели создана модель базы данных в PostgreSQL 9.3, представленная на рис. .

|  |
| --- |
| data.jpg  Рис. 3 Даталогическая модель предметной области в СУБД PostgreSQL 9.3 |

### Разработка структуры программного изделия

Структура системы состоит из нескольких подсистем, которые обеспечивают ее правильное функционирование, как показано на рисунке 4.

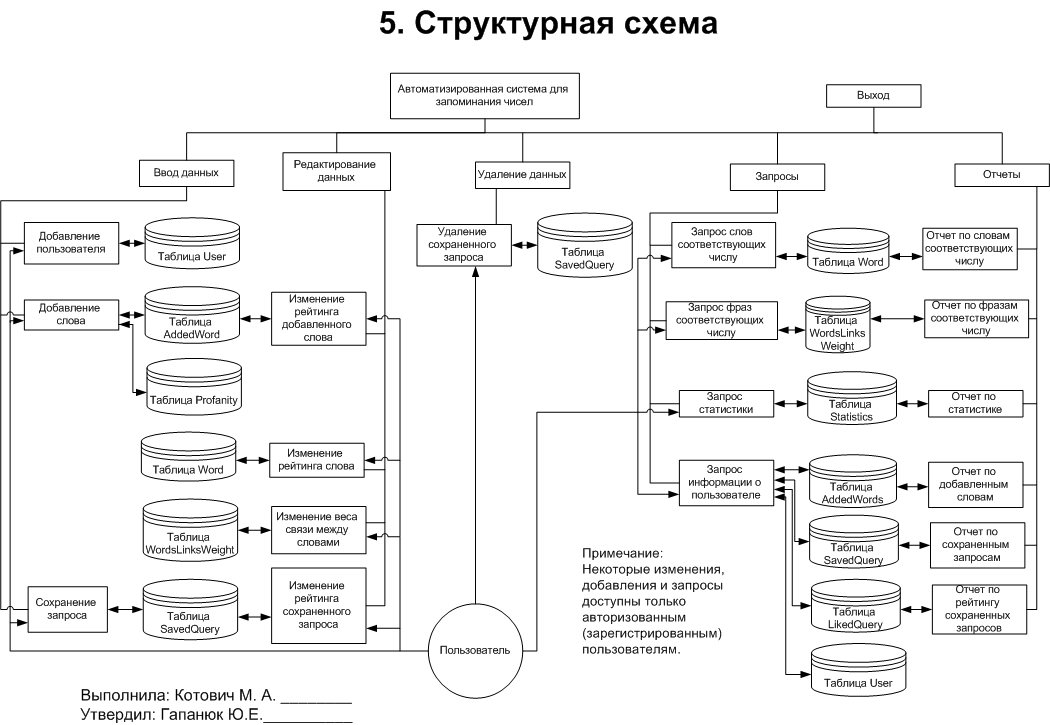


Рис. 4 Структурная схема

### Особенности выбранных технологий

Автоматизированная система для запоминания чисел использует следующие технологии: язык программирования Python 3.4, фреймворк Django 1.8, СУБД PostgreSQL 9.3, web-серверы nginx и uwsgi, СУБД Redis, фреймворк Twitter Bootstrap 2, библиотека JQuery.

Для разработки использовалась IDE Pycharm и система управления версиями Git.

#### Язык программирования Python 3.4

В качестве языка программирования был выбран Python. Python – это интерпретируемый язык программирования. Интерпретируемость языка делает отладку программ проще, однако из-за этого сравнительно понижается скорость выполнения. [10] Но это не является критичным для нересурсоемких задач, которые решает Автоматизированная система для запоминания чисел. Простой и понятный синтаксис Python’а сокращает время разработки приложения. В Python предусмотрена динамическая типизация. [11] Это значит, что не надо заранее объявлять тип переменной. И данную особенность удобно использовать при написании программ. Программа, написанная на Python, будет функционировать одинаково вне зависимости от того, в какой операционной системе она запущена. [8]

Вот лишь некоторые вещи, которые умеет делать python:

* Работа с xml/html файлами
* Работа с http запросами
* GUI (графический интерфейс)
* Создание веб-сценариев
* Работа с FTP
* Работа с изображениями, аудио и видео файлами
* Робототехника
* Программирование математических и научных вычислений

В Python 3.4 были произведены следующие дополнения по сравнению с 3.3:

1. В поставку добавлен инсталлятор для пакетного менеджера pip;
2. Новые файловые дескрипторы теперь по умолчанию не наследуются дочерними подпроцессами;
3. Опция командной строки для изолированного режима;
4. Улучшения в обработке нетекстовых кодеков;
5. Стандартизирован тип "ModuleSpec" для предоставления метаданных системы импорта модулей на стадии до непосредственной загрузки модуля.

#### Фреймворк Django 1.8

Django – это свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC.

Django предоставляет абстрактный уровень (“модели”) для создания структуры и управления данными web-приложения и использует “представления” для инкапсуляции логики обработки запроса и ответа на этот запрос. Шаблоны предоставляют удобный синтаксис для управления отображением информации пользователю. Фреймворк предоставляет мощный инструмент для создания форм и обработки данных, полученных с их помощью. [9]

В составе Django есть собственный web-сервер для разработки. Сервер мониторит изменения в файлах исходного кода автоматически и перезапускается. Так значительно ускоряется процесс разработки на Python, но это пригодно только для разработки и отладки приложения. [12, 16]

Некоторые возможности Django:

* API доступа к БД с поддержкой транзакций;
* Диспетчер URL на основе регулярных выражений;
* Расширяемая система шаблонов с тегами и наследованием;
* Система кэширования.

#### СУБД PostgreSQL 9.3

PostgreSQL - это полноценная SQL СУБД с большим списком возможностей и огромным количеством людей по всему миру, которые используют и разрабатывают эту СУБД. Разработка PostgreSQL ориентировалась на использование в сложных приложениях, поэтому упор всегда делался на надёжность, наличие развитой функциональности и соответствие стандартам. При этом PostgreSQL можно точно также использовать и в веб-приложениях, где данная СУБД показывает неизменно отличные результаты, при лучшей масштабируемости и настраиваемости.

PostgreSQL - это объектно-реляционная система управления базами данных (ORDBMS, СУБД) основанная на POSTGRES, Версии 4.2, которая была разработана в Научном Компьютерном Департаменте Беркли Калифорнийского Университета. Слово "объект-реляционная" означает возможность использования части функциональности, связанной с объектно-ориентированным программированием, например, такой как наследование.

PostgreSQL является прямым потомком с открытым исходным кодом от оригинального кода, сделанного в Беркли. PostgreSQL поставляется под лицензией BSD, что обеспечивает максимальную открытость и доступность.

PostgreSQL является кроссплатформенным продуктом и работает не только в широком диапазоне диалектов UNIX (Linux, FreeBSD, Solaris и т.д.), но и на платформе Microsoft Windows.

Для работы с PostgreSQL существует множество интерфейсов и библиотек взаимодействия из других языков программирования: Java (JDBC), ODBC, Perl, Python, Ruby, C, C++, PHP, Lisp, Scheme и Qt.

Кроме стандартного набора типов, PostgreSQL предоставляет возможность создания списков (тип ENUM), массивов типов, составных типов наподобие структур в языке C, а также имеет типы для уникальной идентификации объектов (OID) и псевдотипы для хранимых процедур. [17]

В версии 9.3 были произведены следующие улучшения и новшества:

* материализированные представления;
* обновляемые представления;
* триггеры к событиям;
* рекурсивные представления;
* латеральное присоединение;
* изменяемые внешние таблицы;
* функции и операторы для работы с типом JSON.

#### Web-сервер Nginx

Nginx – это веб-сервер и почтовый прокси-сервер, работающий на Unix-подобных операционных системах. Начиная с версии 0.7.52 появилась экспериментальная сборка под Microsoft Windows.

В nginx рабочие процессы обслуживают единовременно множество соединений, выполняя цикл обработки событий от дескрипторов. Данные, полученные от клиента, разбираются посредством конечного автомата. Разобранный запрос последовательно обрабатывается цепочкой модулей, задаваемой конфигурацией. Ответ для клиента формируется в буферах, которые хранят данные или в памяти, или указывают на отрезок файла. Буферы объединяются в цепочки, задающие последовательность, в которой данные будут отданы клиенту. [6]

Конфигурация HTTP-сервера nginx разделяется на виртуальные серверы. Виртуальные серверы разделяются на location’ы (location).

Для осуществления эффективного управления памятью в nginx используются пулы. Пулом является последовательность предварительно выделенных блоков динамической памяти. Длина блока обычно от 1 до 16 килобайт.

В Nginx есть модуль географической классификации клиентов по IP-адресу. В его основу входит база данных соответствия IP-адресов географическому региону. [13]

Основные функции сервера:

* Обслуживание статических запросов, индексных файлов, автоматическое создание списка файлов, кэш дескрипторов открытых файлов;
* Акселерированное обратное проксирование с кэшированием, распределение нагрузки и отказоустойчивость;
* Акселерированная поддержка FastCGI, uwsgi, SCGI и memcached серверов с кэшированием, распределение нагрузки и отказоустойчивость;
* Модульность, фильтры, в том числе сжатие (gzip), byte-ranges (докачка), chunked ответы, XSLT-фильтр, SSI-фильтр, преобразование изображений; несколько подзапросов на одной странице, обрабатываемые в SSI-фильтре через прокси или FastCGI/uwsgi/SCGI, выполняются параллельно;
* Поддержка SSL и расширения TLS SNI;
* Поддержка HTTP/2 с приоритизацией на основе весов и зависимостей.

Другие возможности сервера:

* Виртуальные серверы, определяемые по IP-адресу и имени;
* Поддержка keep-alive и pipelined соединений;
* Гибкость конфигурации;
* Изменение настроек и обновление исполняемого файла без перерыва в обслуживании клиентов;
* Настройка форматов логов, буферизованная запись в лог, быстрая ротация логов, запись в syslog;
* Специальные страницы для ошибок 3xx-5xx;
* rewrite-модуль: изменение URI с помощью регулярных выражений;
* Выполнение разных функций в зависимости от адреса клиента;
* Ограничение доступа в зависимости от адреса клиента, по паролю (HTTP Basic аутентификация) и по результату подзапроса;
* Проверка HTTP referer;
* Методы PUT, DELETE, MKCOL, COPY и MOVE;
* FLV и MP4 стриминг;
* Ограничение скорости отдачи ответов;
* Ограничение числа одновременных соединений и запросов с одного адреса;
* Встроенный Perl.

#### Web-сервер UWSGI

**WSGI** – это стандарт взаимодействия между Python-программой, выполняющейся на стороне сервера, и самим веб-сервером. WSGI предоставляет простой и универсальный интерфейс между большинством веб-серверов и веб-приложениями или фреймворками, а uWSGI — это WSGI-сервер.

**CGI** (от англ. Common Gateway Interface — «общий интерфейс шлюза») — стандарт интерфейса, используемого для связи внешней программы с веб-сервером. Программу, которая работает по такому интерфейсу совместно с веб-сервером, принято называть шлюзом, хотя многие предпочитают названия «скрипт» (сценарий) или «CGI-программа».

Сам интерфейс разработан таким образом, чтобы можно было использовать любой язык программирования, который может работать со стандартными устройствами ввода-вывода. Такими возможностями обладают даже скрипты для встроенных командных интерпретаторов операционных систем, поэтому в простых случаях могут использоваться даже командные скрипты.

CGI является одним из наиболее распространённых средств создания динамических сайтов.

Интерфейс **FastCGI** — клиент-серверный протокол взаимодействия веб-сервера и приложения, дальнейшее развитие технологии CGI. По сравнению с CGI является более производительным и безопасным.

FastCGI снимает множество ограничений CGI-программ. Недостаток CGI-программ в том, что они должны быть перезапущены веб-сервером при каждом запросе, что приводит к понижению производительности. FastCGI, вместо того чтобы создавать новые процессы для каждого нового запроса, использует постоянно запущенные процессы для обработки множества запросов. Это позволяет экономить время.

**Simple Common Gateway Interface (SCGI)** — протокол по взаимодействию приложений с веб (http) серверами, разработанный как альтернатива Common Gateway Interface. Он похож на FastCGI, но проще в реализации.

Было принятно решение использовать протокол WSGI, так как он совершеннее CGI и его функционала достаточно для решения задачи реализации автоматизированной системы. Следовательно, вместе с выбранным протоколом, используется и соответствующий сервер uWSGI.

Этот сервер быстрый, потому что написан на C, удобный, потому что легко конфигурируется (.ini, yaml, json, XML или через переменные окружения), надежный, потому что имеет функцию самоуправления: если запрос длится дольше отведенного времени, то master-процесс завершит исполнение текущего worker'a, чтобы не забивать память. [19]

#### СУБД Redis

Redis — это сетевое журналируемое хранилище данных типа «ключ — значение» с открытым исходным кодом. Является нереляционной высокопроизводительной СУБД.

Хранит базу данных в оперативной памяти, снабжена механизмами снимков и журналирования для обеспечения постоянного хранения (на диске). Также предоставляет операции для реализации механизма обмена сообщениями в паттерне Издатель-подписчик. С его помощью приложения могут создавать каналы, подписываться на них и помещать в каналы сообщения, которые будут получены всеми подписчиками (как IRC-чат). Поддерживает репликацию данных с основных узлов на несколько подчинённых (англ. master — slave replication). Также Redis поддерживает транзакции и пакетную обработку команд (выполнение пакета команд, получение пакета результатов).

Redis работает на большинстве POSIX систем, таких как Linux, без каких-либо дополнений. Linux и Mac OS X — две операционные системы, в которых был разработан и в большей степени протестирован Redis, поэтому VMware рекомендует использовать именно их для развертывания. Официальной поддержки для сборок Windows нет, но доступны некоторые опции, позволяющие обеспечить работу Redis на этой ОС. Компания Microsoft активно работает над переносом Redis на Windows. [18]

Все данные Redis хранит в виде словаря, в котором ключи связаны со своими значениями. Одно из ключевых отличий Redis от других хранилищ данных заключается в том, что значения этих ключей не ограничиваются строками. Поддерживаются следующие абстрактные типы данных:

* Строки
* Списки
* Множества
* Хеш-таблицы
* Упорядоченные множества

Тип данных значения определяет, какие операции (команды) доступны для него. Redis поддерживает такие высокоуровневые операции, как объединение и разность наборов, а также их сортировку.

Высокая производительность Redis обуславливается тем, что все данные хранятся в оперативной памяти. На Linux-сервере начального уровня был установлен результат в 110 000 запросов SET и 81 000 запросов GET в секунду.

#### Фреймворк Twitter Bootstrap

Bootstrap (также известен как Twitter Bootstrap)— свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML и CSS шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения. [15]

Bootstrap использует самые современные наработки в области CSS и HTML.

Основные преимущества Bootstrap 2:

* Экономия времени — Bootstrap позволяет сэкономить время и усилия, используя шаблоны дизайна и классы, и сконцентрироваться на других разработках;
* Высокая скорость — динамичные макеты Bootstrap масштабируются на разные устройства и разрешения экрана без каких-либо изменений в разметке;
* Гармоничный дизайн — все компоненты платформы Bootstrap используют единый стиль и шаблоны с помощью центральной библиотеки. Дизайн и макеты веб-страниц согласуются друг с другом;
* Простота в использовании — платформа проста в использовании, пользователь с базовыми знаниями HTML и CSS может начать разработку с Twitter Bootstrap;
* Совместимость с браузерами — Twitter Bootstrap совместим с Mozilla Firefox, Yandex Browser, Google Chrome, Safari, Internet Explorer, Microsoft Edge и Opera;
* Открытое программное обеспечение — особенность Twitter Bootstrap, которая предполагает удобство использования, посредством открытости исходных кодов и бесплатной загрузки.

#### Библиотека JQuery

jQuery — библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX.

Точно так же, как CSS отделяет визуализацию от структуры HTML, JQuery отделяет поведение от структуры HTML. Например, вместо прямого указания на обработчик события нажатия кнопки управление передаётся JQuery, которая идентифицирует кнопки и затем преобразует его в обработчик события клика. Такое разделение поведения и структуры также называется принципом ненавязчивого JavaScript.

Библиотека jQuery содержит функциональность, полезную для максимально широкого круга задач. Тем не менее, разработчиками библиотеки не ставилась задача совмещения в jQuery функций, которые подошли бы всюду, поскольку это привело бы к большому коду, бо́льшая часть которого не востребована. Поэтому была реализована архитектура компактного универсального ядра библиотеки и плагинов. Это позволяет собрать для ресурса именно ту JavaScript-функциональность, которая на нём была бы востребована.

Основные возможности библиотеки:

* Движок кроссбраузерных CSS-селекторов Sizzle, выделившийся в отдельный проект;
* Переход по дереву DOM, включая поддержку XPath как плагина;
* События;
* Визуальные эффекты;
* AJAX-дополнения;
* JavaScript-плагины.

#### IDE Pycharm

Для разработки системы была выбрана среда разработки Pycharm. PyCharm — интегрированная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов и поддерживает веб-разработку на Django. PyCharm разработана компанией JetBrains на основе IntelliJ IDEA.

PyCharm работает под операционными системами Windows, Mac OS X и Linux.

Некоторые возможности этой среды разработки:

* + Статический анализ кода, подсветка синтаксиса и ошибок.
  + Навигация по проекту и исходному коду: отображение файловой структуры проекта, быстрый переход между файлами, классами, методами и использованиями методов.
  + Рефакторинг: переименование, извлечение метода, введение переменной, введение константы, подъём и спуск метода и т. д.
  + Инструменты для веб-разработки с использованием фреймворка Django
  + Встроенный отладчик для Python
  + Встроенные инструменты для юнит-тестирования
  + Разработка с использованием Google App Engine
* Поддержка систем контроля версий: общий пользовательский интерфейс для Mercurial, Git, Subversion, Perforce и CVS с поддержкой списков изменений и слияния

#### Система управления версиями Git

Git — мощная и сложная распределенная система контроля версий. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года.

Система спроектирована как набор программ, специально разработанных с учётом их использования в скриптах. Это позволяет удобно создавать специализированные системы контроля версий на базе Git или пользовательские интерфейсы. Например, Cogito является именно таким примером оболочки к репозиториям Git, а StGit использует Git для управления коллекцией исправлений (патчей).

Git поддерживает быстрое разделение и слияние версий, включает инструменты для визуализации и навигации по нелинейной истории разработки. Как и Darcs, BitKeeper, Mercurial, Bazaar и Monotone, Git предоставляет каждому разработчику локальную копию всей истории разработки, изменения копируются из одного репозитория в другой.

Удалённый доступ к репозиториям Git обеспечивается git-daemon, SSH- или HTTP-сервером. TCP-сервис git-daemon входит в дистрибутив Git и является наряду с SSH наиболее распространённым и надёжным методом доступа. Метод доступа по HTTP, несмотря на ряд ограничений, очень популярен в контролируемых сетях, потому что позволяет использовать существующие конфигурации сетевых фильтров.

Ядро Git представляет собой набор утилит командной строки с параметрами. Все настройки хранятся в текстовых файлах конфигурации. Такая реализация делает Git легко портируемым на любую платформу и даёт возможность легко интегрировать Git в другие системы (в частности, создавать графические git-клиенты с любым желаемым интерфейсом).

Репозиторий Git представляет собой каталог файловой системы, в котором находятся файлы конфигурации репозитория, файлы журналов, хранящие операции, выполняемые над репозиторием, индекс, описывающий расположение файлов и хранилище, содержащее собственно файлы. Структура хранилища файлов не отражает реальную структуру хранящегося в репозитории файлового дерева, она ориентирована на повышение скорости выполнения операций с репозиторием. Когда ядро обрабатывает команду изменения (неважно, при локальных изменениях или при получении патча от другого узла), оно создаёт в хранилище новые файлы, соответствующие новым состояниям изменённых файлов. Существенно, что никакие операции не изменяют содержимого уже существующих в хранилище файлов.

По умолчанию репозиторий хранится в подкаталоге с названием «.git» в корневом каталоге рабочей копии дерева файлов, хранящегося в репозитории. Любое файловое дерево в системе можно превратить в репозиторий git, отдав команду создания репозитория из корневого каталога этого дерева (или указав корневой каталог в параметрах программы). Репозиторий может быть импортирован с другого узла, доступного по сети. При импорте нового репозитория автоматически создаётся рабочая копия, соответствующая последнему зафиксированному состоянию импортируемого репозитория (то есть не копируются изменения в рабочей копии исходного узла, для которых на том узле не была выполнена команда commit).

Нижний уровень git является т. н. файловой системой с адресацией по содержимому (content-addressed file system). Инструмент командной строки git содержит ряд команд по непосредственной манипуляции этим репозиторием на низком уровне. Эти команды не нужны при нормальной работе с git как с системой контроля версий, но нужны для реализации сложных операций (ремонт поврежденного репозитория и т. д.), а также дают возможность создать на базе репозитория git какое-то своё приложение.

Для каждого объекта в репозитории вычисляется SHA-1 хеш, и именно он становится именем файла, содержащего данный объект в директории .git/objects. Для оптимизации работы с файловыми системами, не использующими деревья для директорий, первый байт хеша становится именем поддиректории, а остальные — именем файла в ней, что снижает количество файлов в одной директории (лимитирующий фактор производительности на таких устаревших файловых системах).

Все ссылки на объекты репозитория, включая ссылки на один объект, находящийся внутри другого объекта, являются SHA-1 хешами.

Кроме того, в репозитории существует директория refs, которая позволяет задать читаемые человеком имена для каких-то объектов git. В командах git оба вида ссылок — читаемые человеком из refs, и нижележащие SHA-1 — полностью взаимозаменяемы.

В классическом обычном сценарии в репозитории git есть три типа объектов — файл, дерево и коммит. Файл есть какая-то версия какого-то пользовательского файла, дерево — совокупность файлов из разных поддиректорий, коммит — дерево + некая дополнительная информация (например, родительский(е) коммит(ы), а также комментарий).

В репозитории иногда производится сборка мусора, во время которой устаревшие файлы заменяются на «дельты» между ними и актуальными файлами (актуальная версия файла хранится не-инкрементально, инкременты используются только для «шагания» назад), после чего данные «дельты» складываются в один большой файл, к которому строится индекс. Это снижает требования по месту на диске.

Репозиторий git бывает локальный и удаленный. Локальный репозиторий — это поддиректория .git, создается (в пустом виде) командой git init и (в непустом виде с немедленным копированием содержимого родительского удаленного репозитория и простановкой ссылки на родителя) командой git clone.

Практически все обычные операции с системой контроля версий, такие, как коммит и слияние, производятся только с локальным репозиторием. Удаленный репозиторий можно только синхронизировать с локальным как «вверх» (push), так и «вниз» (pull).

Наличие полностью всего репозитория проекта локально у каждого разработчика дает git ряд преимуществ перед SVN. Так, например, все операции, кроме push и pull, можно осуществлять без наличия Интернет-соединения.

Очень мощной возможностью git являются ветви, реализованные куда более полно, чем в SVN. Создать новую ветвь так же просто, как и совершить коммит. По сути, ветвь git есть не более чем читаемое человеком имя, «навешенное» на некий коммит в репозитории (используется поддиректория refs). Коммит без создания новой ветви всего лишь передвигает эту ссылку на себя, а коммит с созданием ветви — оставляет старую ссылку на месте, но создает новую на новый коммит, и объявляет её текущей. Заменить локальные девелоперские файлы на набор файлов из иной ветви, тем самым перейдя к работе с ней — также тривиально.

Также поддерживаются суб-репозитории с синхронизацией текущих ветвей в них.

Команда push передает все новые данные (те, которых ещё нет в удаленном репозитории) из локального репозитория в репозиторий удаленный. Для исполнения этой команды необходимо, чтобы удаленный репозиторий не имел новых коммитов в себя от других клиентов, иначе push завершается ошибкой, и придётся делать pull и слияние.

Команда pull — обратна команде push. В случае, если одна и та же ветвь имеет независимую историю развития от какого-то момента в локальной и в удаленной копии, pull немедленно переходит к слиянию.

Слияние в пределах разных файлов осуществляется автоматически (все это поведение настраивается), а в пределах одного файла — стандартным трёхпанельным сравнением файлов. После слияния нужно объявить конфликты как разрешенные.

Результатом всего этого является новое состояние в локальных файлах у того разработчика, что осуществил слияние. Ему нужно немедленно сделать коммит, при этом в данном объекте коммита в репозитории окажется информация о том, что коммит есть результат слияния двух ветвей и имеет два родительских коммита.

Кроме слияния, git поддерживает ещё и rebase. Эта операция есть получение набора всех изменений в ветви А, с последующим их «накатом» на ветвь B. В результате ветвь B продвигается до состояния AB. В отличие от слияния, в истории ветви AB не останется никаких промежуточных коммитов ветви A (только история ветви B и запись о самом rebase, это упрощает интеграцию крупных и очень крупных проектов).

Также git имеет временный локальный индекс файлов. Это — промежуточное хранилище между собственно файлами и очередным коммитом (коммит делается только из этого индекса). С помощью этого индекса осуществляется добавление новых файлов (git add добавляет их в индекс, они попадут в следующий коммит), а также коммит НЕ ВСЕХ измененных файлов (коммит делается только тем файлам, которым был сделан git add). После git add можно редактировать файл далее, получатся три копии одного и того же файла — последняя, в индексе (та, что была на момент git add), и в последнем коммите.

Часто называемые преимущества git перед другими DVCS:

* Высокая производительность.
* Развитые средства интеграции с другими VCS, в частности, с CVS, SVN и Mercurial. Помимо разнонаправленных конвертеров репозиториев, имеющиеся в комплекте программные средства позволяют разработчикам использовать git при размещении центрального репозитория в SVN или CVS, кроме того, git может имитировать cvs-сервер, обеспечивая работу через клиентские приложения и поддержку в средах разработки, специально не поддерживающих git.
* Продуманная система команд, позволяющая удобно встраивать git в скрипты.
* Качественный веб-интерфейс «из коробки».
* Репозитории git могут распространяться и обновляться общесистемными файловыми утилитами архивации и обновления, такими как rsync, благодаря тому, что фиксации изменений и синхронизации не меняют существующие файлы с данными, а только добавляют новые (за исключением некоторых служебных файлов, которые могут быть автоматически обновлены с помощью имеющихся в составе системы утилит). Для раздачи репозитория по сети достаточно любого веб-сервера.

В числе недостатков git обычно называют:

* Отсутствие сквозной нумерации коммитов монотонно непрерывно возрастающими целыми числами. Во многих проектах используется автоматические получение номера этой версии (например, командой svnversion), построение .H файла на основе этого числа, и далее его использование при создании штампа версии исполняемого файла, некоторых вшитых в него строк и так далее.
* Отсутствие переносимой на другие операционные системы поддержки путей в кодировке Unicode в Microsoft Windows (для версий msysgit до 1.8.1). Если путь содержит символы, отличные от ANSI, то их поддержка из командной строки требует специфических настроек, которые не гарантируют правильного отображения файловых имён при пользовании тем же репозиторием из других ОС. Одним из способов решения проблемы для git 1.7 является использование специально пропатченного консольного клиента. Другой вариант — использование графических утилит, работающих напрямую через API, таких как TortoiseGit.
* Некоторое неудобство для пользователей, переходящих с других VCS. Команды git, ориентированные на наборы изменений, а не на файлы, могут вызвать недоумение у пользователей, привыкших к файл-ориентированным VCS, таким как SVN. Например, команда «add», которая в большинстве систем управления версиями производит добавление файла к проекту, в git подготавливает к фиксации сделанные в файлах изменения. При этом сохраняется не патч, описывающий изменения, а новая версия целевого файла.
* Использование для идентификации ревизий хешей SHA1, что приводит к необходимости оперировать длинными строками вместо коротких номеров версий, как во многих других системах (хотя в командах допускается использование неполных хеш-строк).
* Бо́льшие накладные расходы при работе с проектами, в которых делаются многочисленные несвязанные между собой изменения файлов. При работе в таком режиме размеры наборов изменений становятся достаточно велики и происходит быстрый рост объёма репозиториев.
* Бо́льшие затраты времени, по сравнению с файл-ориентированными системами, на формирование истории конкретного файла, истории правок конкретного пользователя, поиска изменений, относящихся к заданному месту определённого файла.
* Отсутствие отдельной команды переименования/перемещения файла, которая отображалась бы в истории как соответствующее единое действие. Существующий скрипт git mv фактически выполняет переименование, копирование файла и удаление его на старом месте, что требует специального анализа для определения, что в действительности файл был просто перенесён (этот анализ выполняется автоматически командами просмотра истории). Однако, учитывая тот факт, что наличие специальной команды для переименования/перемещения файлов технически не вынуждает пользователя использовать именно её (и, как следствие, в этом случае возможны разрывы в истории), поведение git может считаться преимуществом.
* Система работает только с файлами и их содержимым, и не отслеживает пустые каталоги.

Несмотря на большое количество недостатков, достоинства системы git превалируют над ними. Поэтому для удобства работы над интеллектуальной системой, для удобного и своевременного создания резервных копий и хранения прошлых версий проекта, была выбрана система git.

# Технологическая часть

## Разработка интерфейса взаимодействия

### Граф диалога

Разработка интерфейса взаимодействия заключается в разработке графа диалога, представленного на рис. 5.

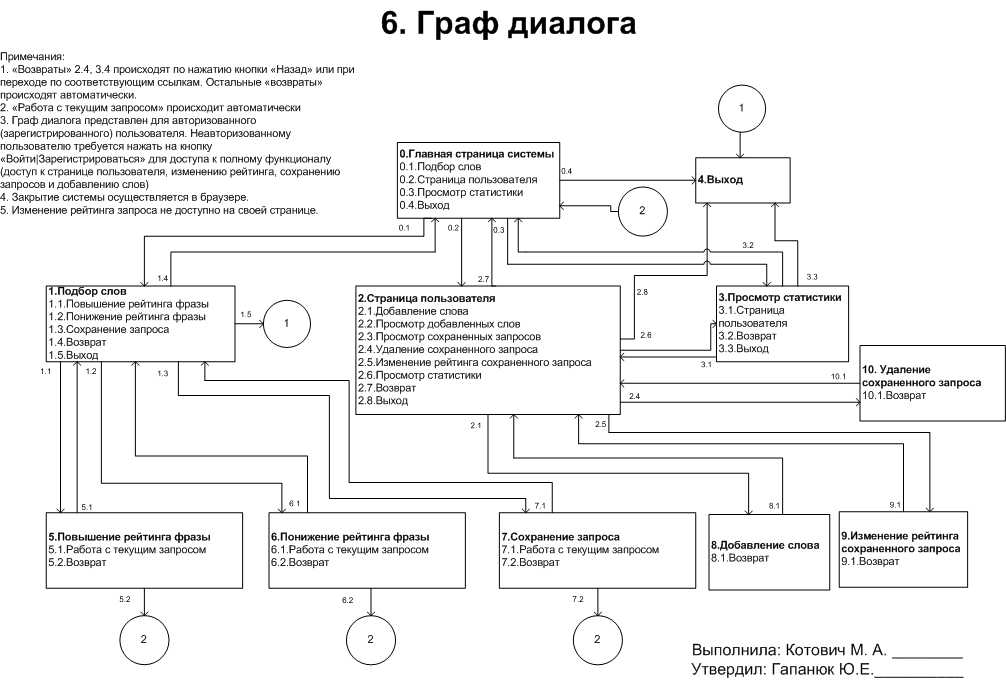


Рисунок 5. Граф диалога

### Интерфейс взаимодействия с пользователем

Интерфейс взаимодействия с пользователем разработан исходя из следующих требований:

1. необходимость регистрации и авторизации пользователя;
2. необходимость добавления слова;
3. необходимость поиска слов;
4. необходимость повышения и понижения рейтинга слов;
5. необходимость сохранения запроса
6. необходимость повышения и понижения рейтинга страницы пользователя;
7. необходимость просмотра статистики;
8. необходимость удаления сохраненных запросов.

Форма регистрации (авторизации) пользователя представлена на рис. 6.

|  |
| --- |
| Рис. 6 Форма регистрации (авторизации) пользователя |

Форма добавления слова представлена на рис.7.1-7.5 .

|  |
| --- |
| Рис. 7.1 Форма добавления слова |

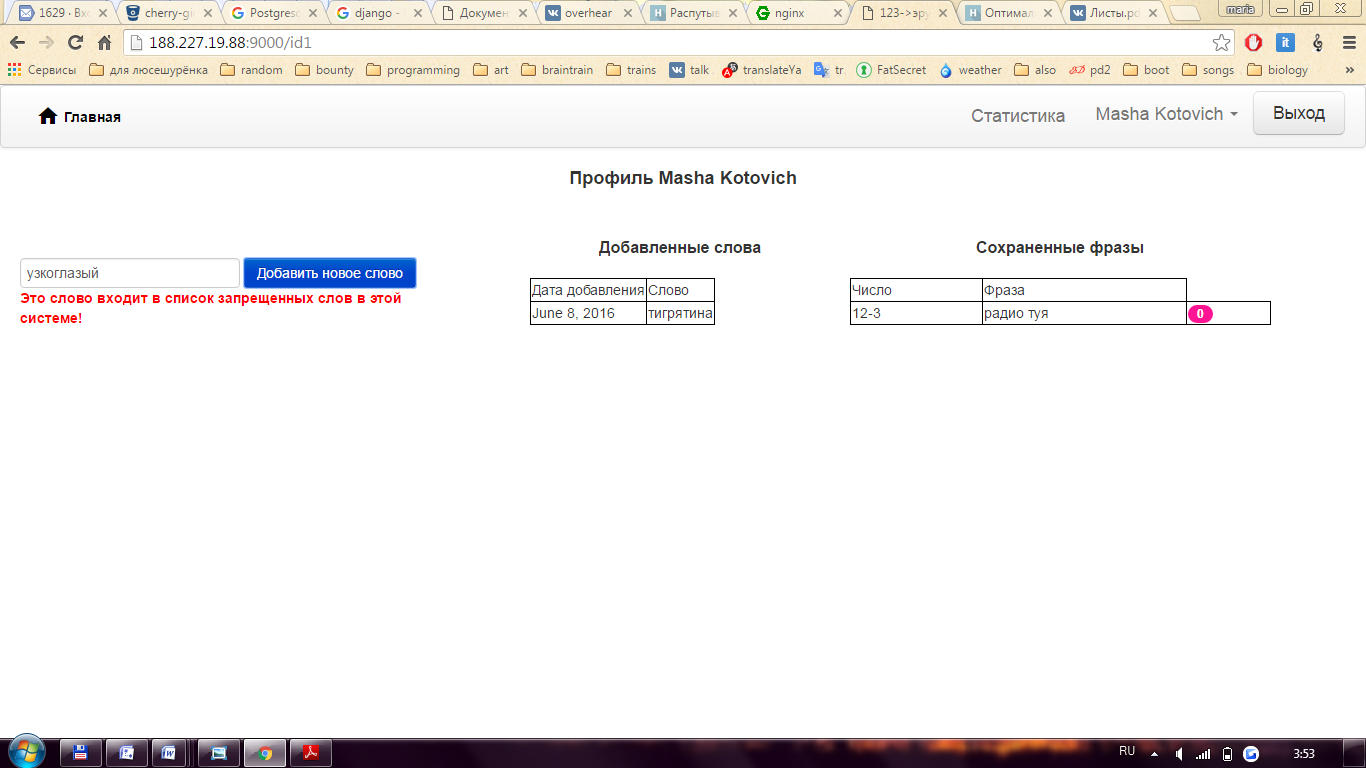


Рис. 7.2 Форма добавления слова

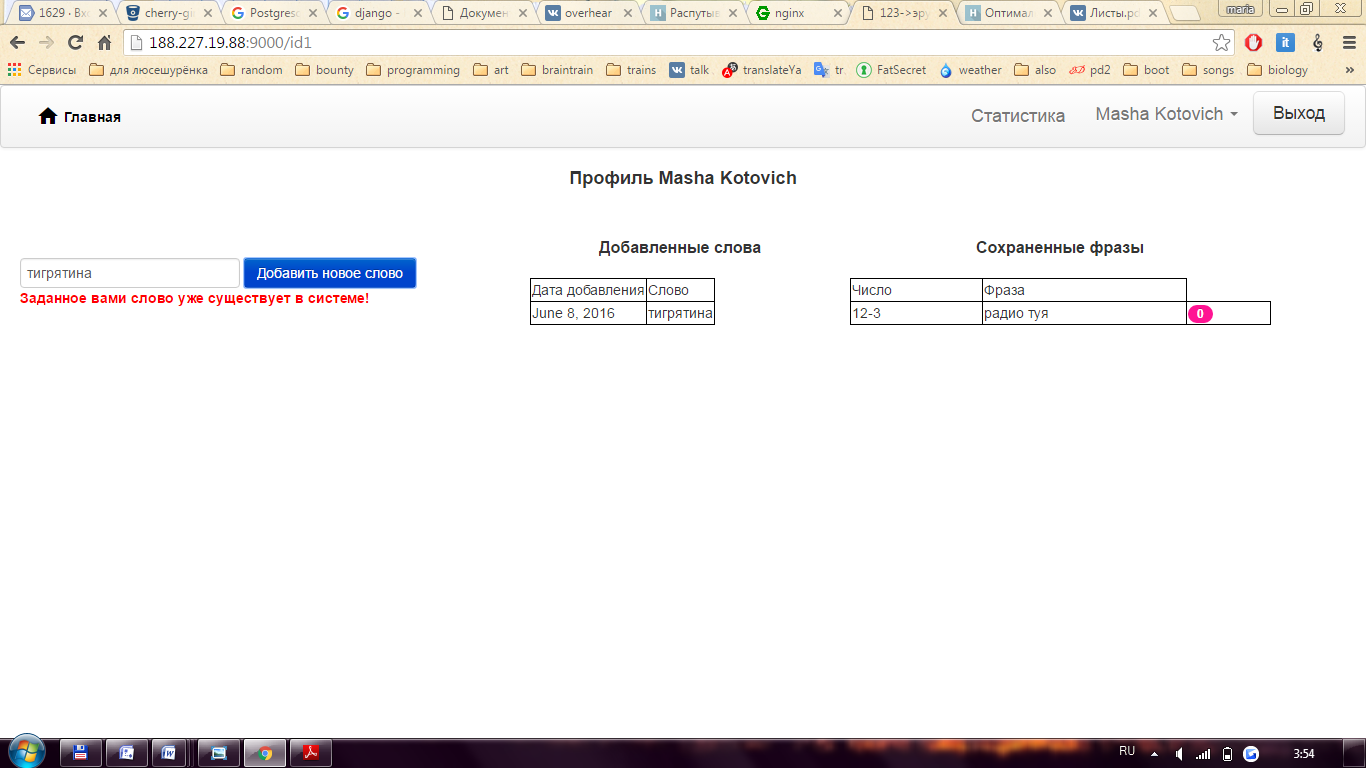


Рис. 7.3 Форма добавления слова

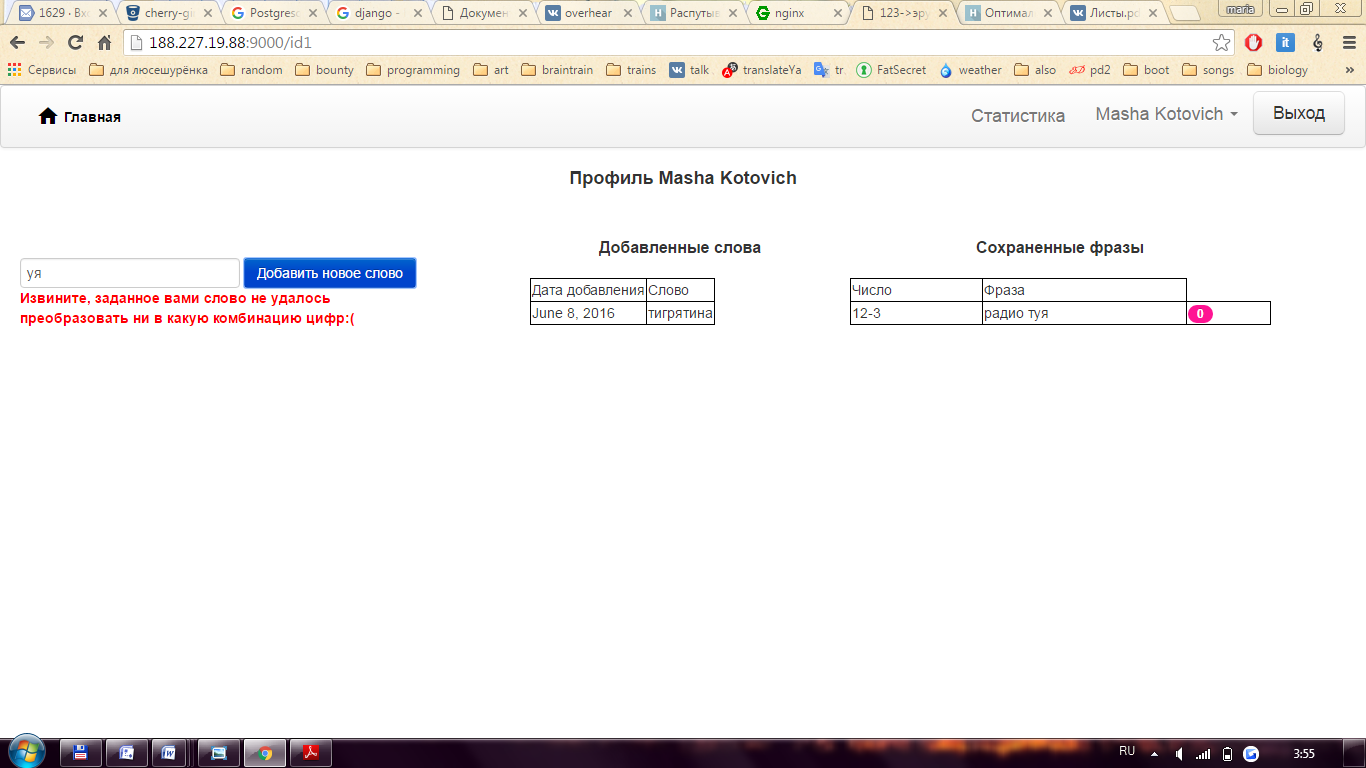


Рис. 7.4 Форма добавления слова

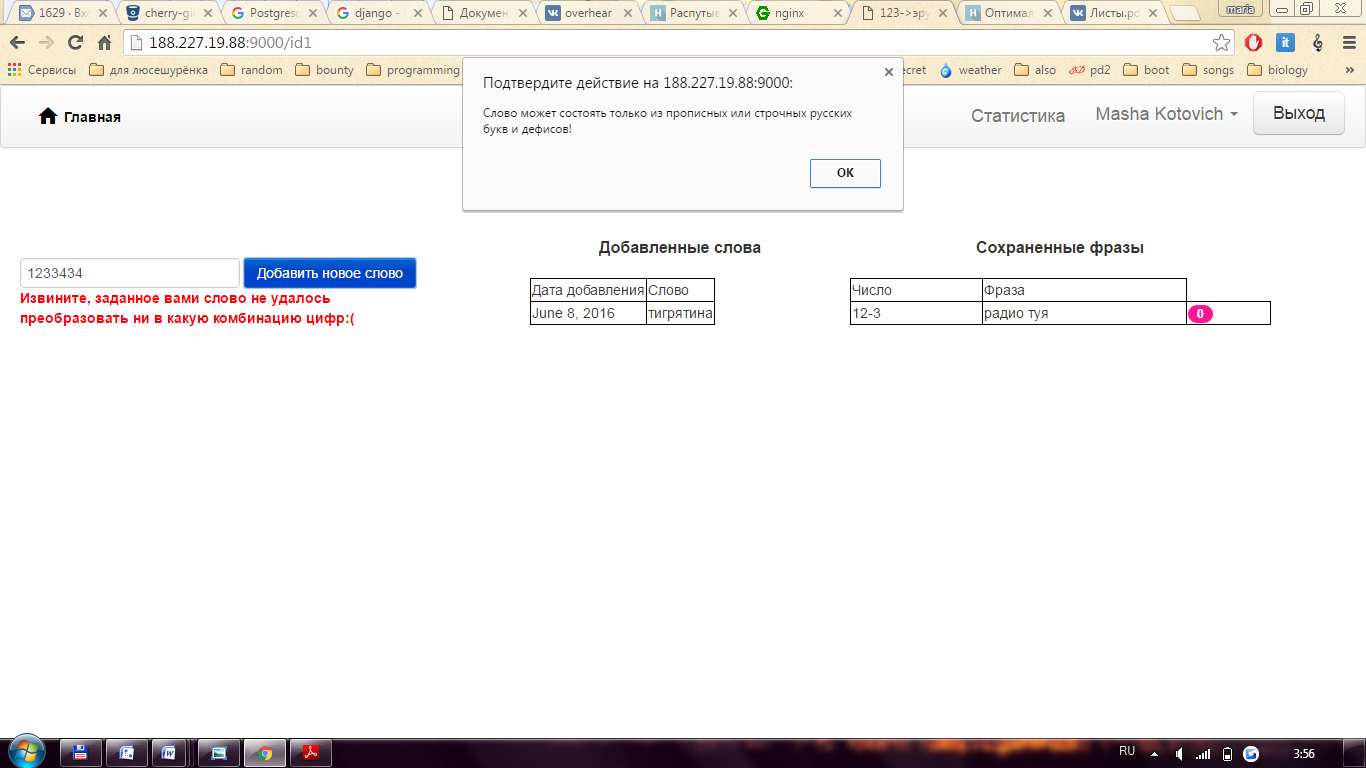


Рис. 7.5 Форма добавления слова. Неверный формат данных.

Форма поиска слов представлена на рис.8.1, 8.2.

|  |
| --- |
| Рис. 8.1 Форма поиска слов |

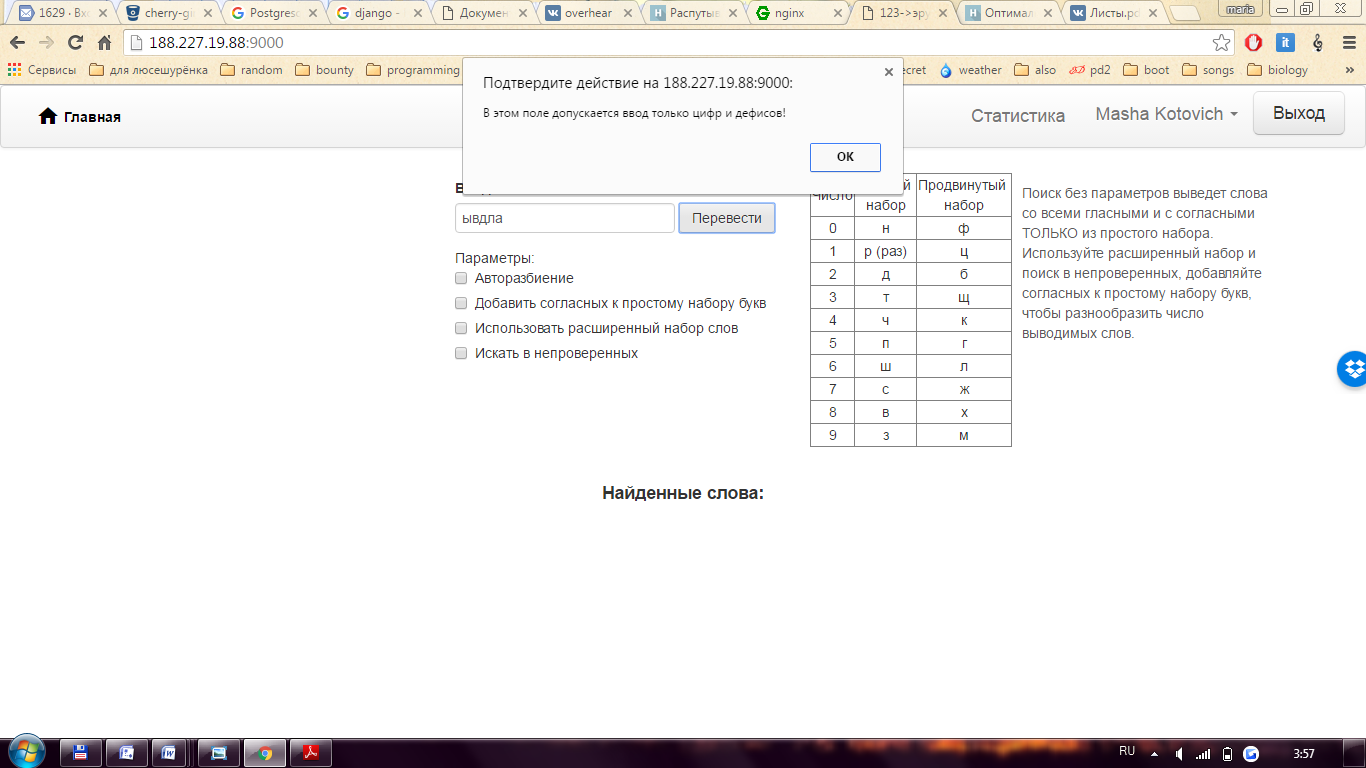


Рис. 8.2 Форма поиска слов. Неверный формат данных.

Форма повышения (понижения) рейтинга слова и сохранения запроса с одним словом представлена на рис.9.

|  |
| --- |
| Рис. 9 Форма повышения (понижения) рейтинга слова и сохранения запроса с одним словом |

Форма повышения (понижения) рейтинга фразы и сохранения запроса представлена на рис. 10.1-10.4.

|  |
| --- |
| Рис. 10.1 Форма повышения (понижения) рейтинга фразы и сохранения запроса |

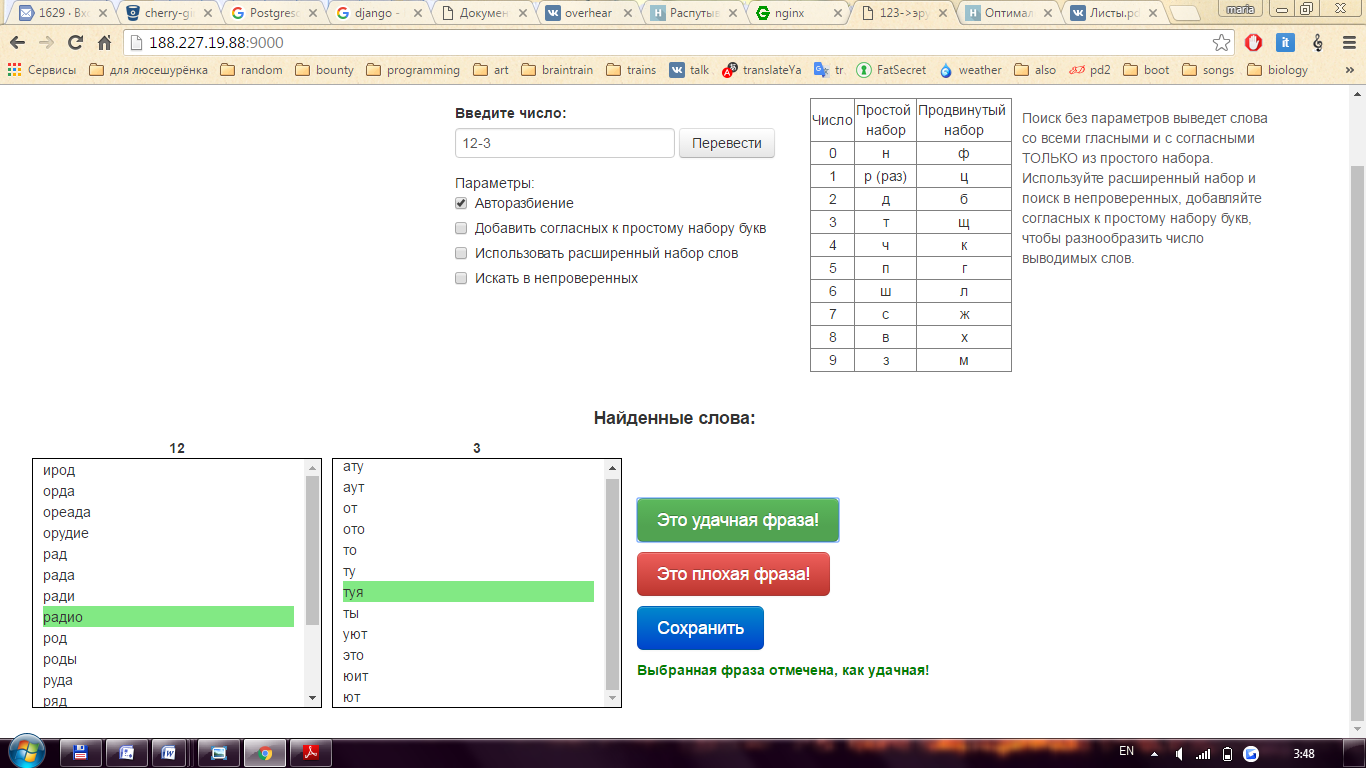


Рис. 10.2 Форма повышения (понижения) рейтинга фразы и сохранения запроса

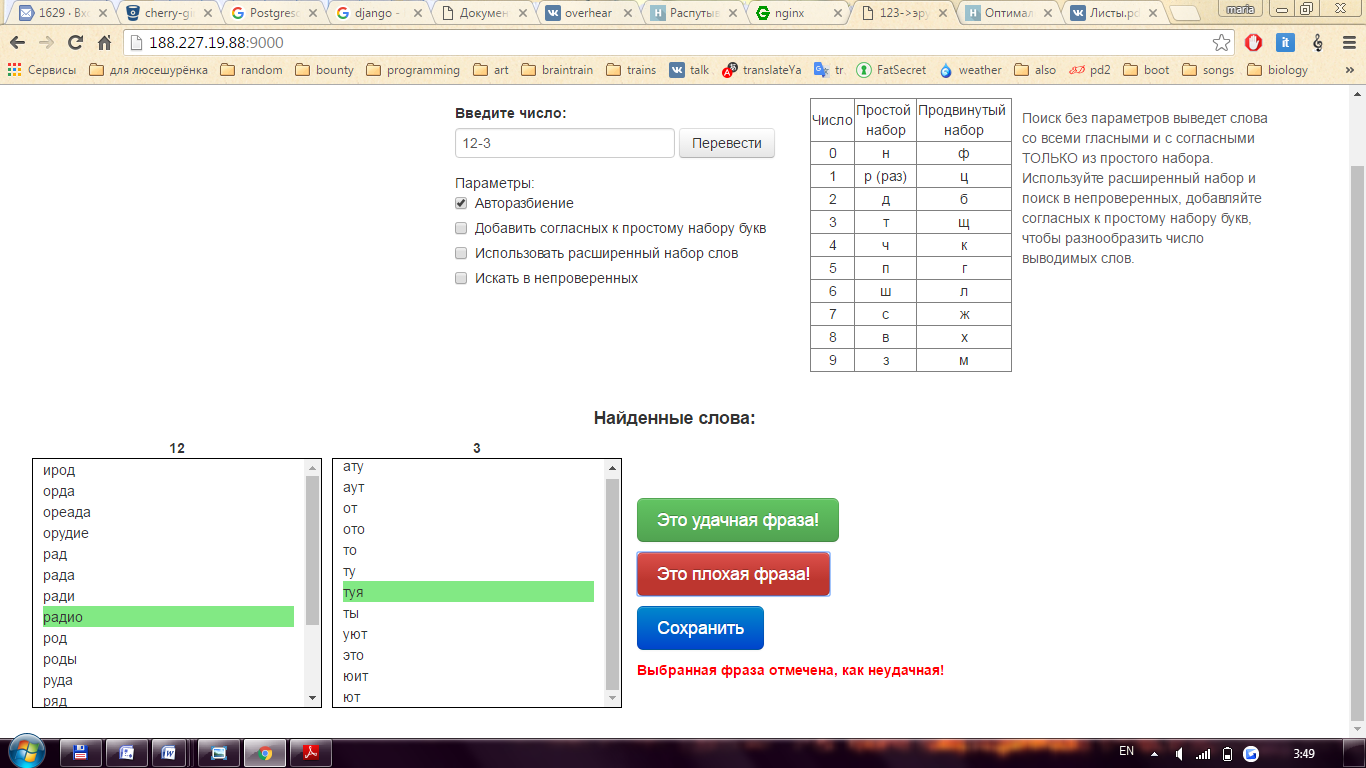


Рис. 10.3 Форма повышения (понижения) рейтинга фразы и сохранения запроса

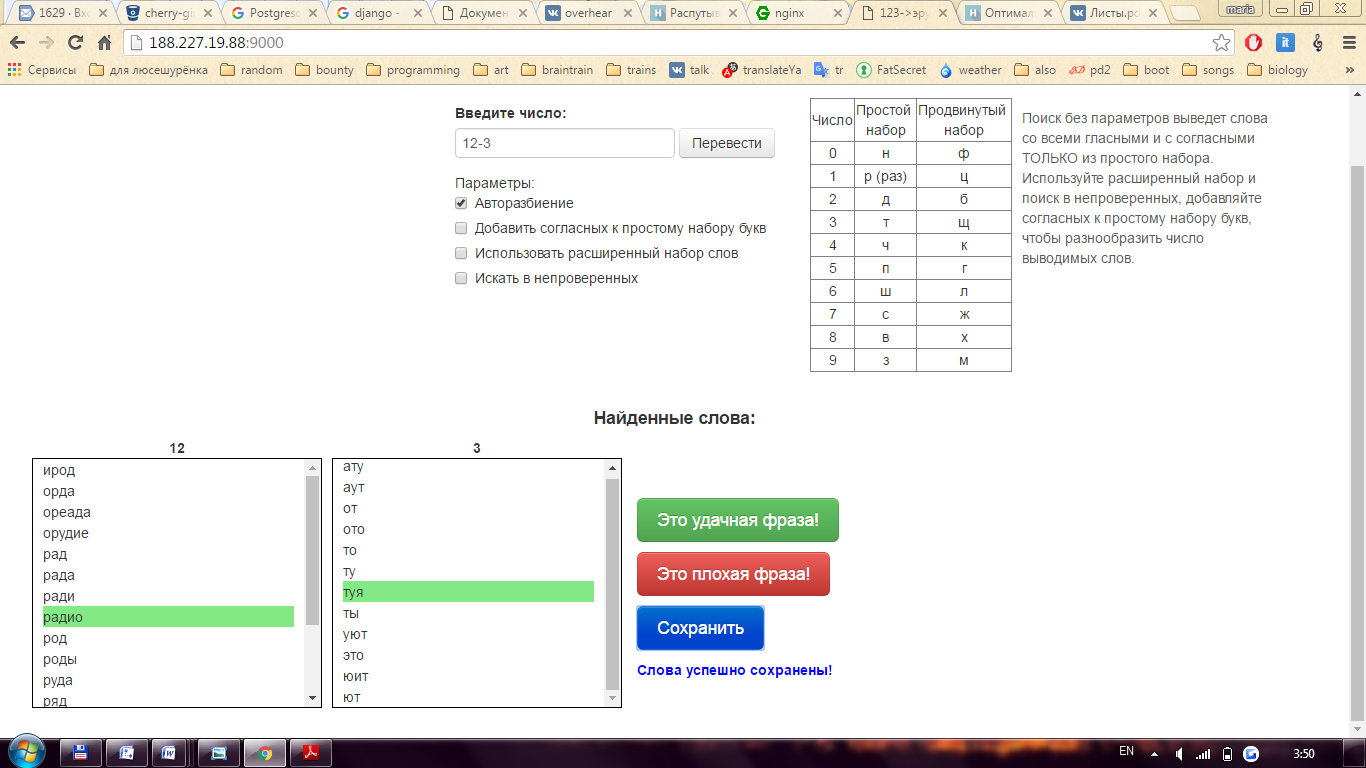


Рис. 10.4 Форма повышения (понижения) рейтинга фразы и сохранения запроса

Форма повышения (понижения) рейтинга страницы пользователя представлена на рис.11.

|  |
| --- |
| Рис. 11 Форма повышения (понижения) рейтинга страницы пользователя |

Статистика представлена на рис. 12.

|  |
| --- |
| Рис. 12 Статистика |

Форма удаления сохраненных запросов представлена на рис. 13.

|  |
| --- |
| Рис. 13 Форма удаления сохраненных запросов |

### Интерфейс взаимодействия с администратором и модератором

Администраторы и модераторы осуществляют свои действия главным образом по добавлению/удалению/редактированию слов и переносу слов из добавленных в основные. Эти действия не являются трудоемкими, поэтому их работа осуществляется в терминале (консоли).

# Научно-исследовательская часть. Выбор алгоритма авторазбиения чисел

## Описание предметной области

При вводе больших чисел в автоматизированную систему пользователю необходимо было бы вручную перебирать возможные варианты разбиения чисел на группы цифр. Это является времязатратной работой, так как не для каждой группы цифр существует соответствующий набор слов. Поэтому в автоматизированную систему необходимо внедрить алгоритм авторазбиения, который поспособствует уменьшению временных затрат пользователя при работе с системой для запоминания чисел.

В связи с поставленной задачей произведем выбор алгоритма авторазбиения из следующих вариантов:

1. Комбинаторный алгоритм

2. Алгоритм случайного разбиения

## Комбинаторный алгоритм

Комбинаторный алгоритм заключается в том, что слова ищутся для каждой группы цифр из множества вариантов разбиения. Число таких вариантов разбиения вычисляется по простой комбинаторной формуле:

(3.1)

Эта формула широко применима во многих предметных областях и во многих задачах. Также она учитывает все множество разбиений. В этом, несомненно, ее плюсы. Однако у данной формулы существует ряд минусов: у больших чисел будут выделяться большие подмножества цифр, не имеющие в соответствии слов. Таких подмножеств будет много, следовательно, немалая часть результатов такого расчета будет бессмысленна для пользователя. Также алгоритм имеет высокую сложность, что повышает время его выполнения. [23]

## Алгоритм случайного разбиения

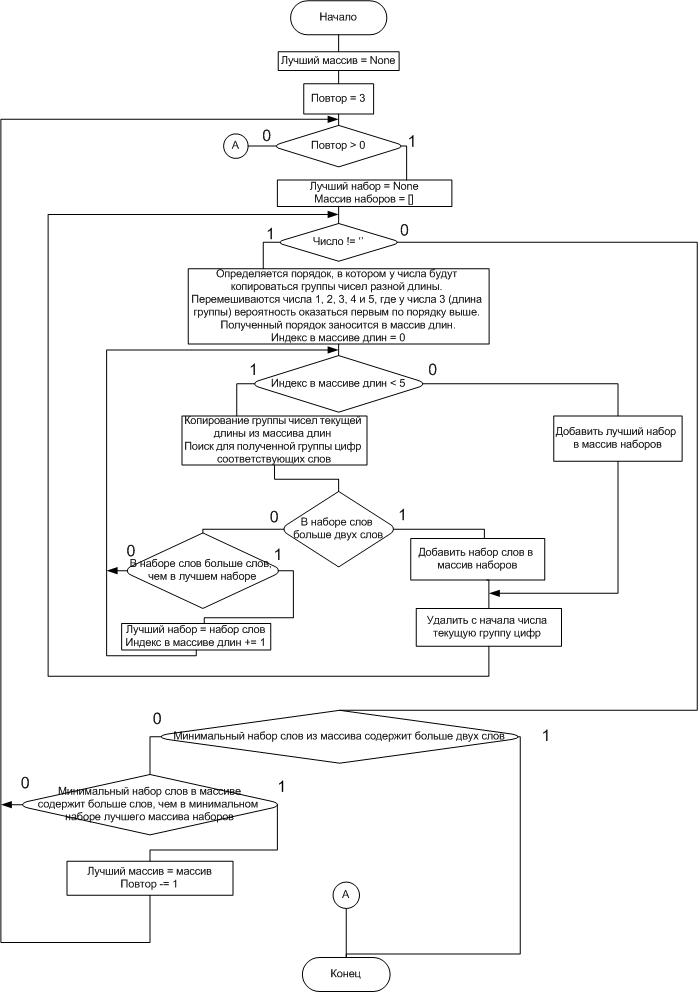
Блок-схема алгоритма случайного разбиения представлена на рисунке 15.

Данный алгоритм никогда не выделит большую группу цифр, так как умеет выделять только группы цифр длиной от 1 до 5 символов. Рассматривая первую итерацию алгоритма можно заметить, что он выделяет массив перемешанных длин от 1 до 5. От исходного числа копируется группа цифр длиной, равной нулевому элементу массива. Если для полученной группы цифр нашлось меньше двух слов, то от исходного числа копируется другая группа цифр (с начала числа) длиной, равной следующему элементу массива и т.д.

Следует отметить, что вероятность попадания на нулевую позицию в массиве у чисел от 1 до 5 разная. У числа 3 вероятность попадания самая высокая, у числа 2 – ниже, и так далее. Это зависит от того, насколько информативными и полезными для пользователя будут слова, содержащие три буквы из цифро-буквенного алфавита, две буквы, четыре, одну и пять. Очевидно, что чисел с тремя согласными из цифро-буквенного алфавита больше, чем с пятью.

Этот алгоритм сложно понять и осознать с первого раза, однако он никогда не будет работать впустую и быстро выдаст хорошие комбинации слов.

В отличие от комбинаторного алгоритма, его применимость к различным задачам гораздо уже. [23]

 Рис. 15 Блок-схема алгоритма случайного разбиения

## Выбор наиболее подходящего алгоритма

Рассмотрев выше два подхода к решению задачи, мы увидели, что у них есть свои определенные плюсы и минусы. Нельзя однозначно сказать, какой из алгоритмов следует использовать в системе, а какой – нет. Поэтому выберем критерии для сравнения этих алгоритмов:

*–* скорость работы. Скорость работы определяется временем выполнения алгоритма своей задачи.

– применимость. Этот критерий показывает, на скольких задачах еще применим этот алгоритм.

- эффективность. Эффективность алгоритма определяется отношением полезных групп цифр (к которым почти наверняка найдется набор слов) ко всем группам цифр, которые «пробегает» алгоритм.

*-* количество разбиений. Этот критерий показывает, сколько разбиений покажет алгоритм за все время своего выполнения.

В таблице 20 показаны критерии, их обозначения и краткое описание описание. В таблице 21 представлена вербально-числовая шкала для этих критериев.

Таблица 20 Критерии для сравнения

|  |  |  |
| --- | --- | --- |
| Критерий | Обозначение | Описание критерия |
| Скорость работы |  | Время работы алгоритма |
| Применимость |  | Число задач, к которым еще применим алгоритм |
| Эффективность |  | Отношение полезных групп цифр ко всем группам цифр |
| Количество разбиений |  | Разбиения за все время выполнения алгоритма |

Таблица 21 Вербально-числовая шкала

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Оценка |  |  |  |  |
| 5 | <1с | Большое число задач | 1 | Всевозможные разбиения |
| 4 | 2-5с | Не очень много задач | 0,9 | Большая часть возможных разбиений |
| 3 | 6-15с | Среднее число задач | 0,8 | Половина возможных разбиений |
| 2 | 16-30с | Мало задач | 0,7 | Меньше половины возможных разбиений |
| 1 | >30с | Единственное применение | <0,6 | Совсем маленькое число из возможных разбиений |

Необходимо произвести анализ не только среди вариантов алгоритмов по различным критериям, но и учесть относительную важность критериев.

Таблица 22 Варианты алгоритмов для авторазбиения

|  |  |
| --- | --- |
| Вариант | Описание |
|  | Комбинаторный алгоритм |
|  | Алгоритм случайного разбиения |

В табл. 22 приведены варианты алгоритмов для авторазбиения.

В табл. 23 приведены значения критериев для первого варианта.

Таблица 23 Значения критериев для первого варианта

|  |  |
| --- | --- |
| Критерий | Значение |
|  | В среднем 3-4 секунды |
|  | Большое число задач |
|  | В среднем 0,6 |
|  | Всевозможные разбиения |

В табл. 24 приведены значения критериев для второго варианта.

Таблица 24 Значения критериев для второго варианта

|  |  |
| --- | --- |
| Критерий | Значение |
|  | В среднем 1 секунда |
|  | Мало задач |
|  | В среднем 0,9 |
|  | Около половины возможных разбиений |

Для исключения неэффективных альтернативных вариантов, т. е. вариантов, которые не лучше других рассматриваемых вариантов по всем локальным критериям, будем использовать метод отбора парето-оптимальных вариантов. Этот подход позволяет уменьшить количество вариантов, подлежащих дальнейшему сравнению. Будем использовать следующий алгоритм:

1) построим таблицу исходных вариантов, подлежащих сравнению. В этой таблице указывают — численное значение -го варианта по -му критерию;

2) построим Парето-оптимальную квадратную таблицу А размером , где — число сравниваемых вариантов. Присвоим диагональным элементам таблицы значения, равные нулю, а элементам — значения , если вариант предпочтительнее варианта по всем критериям, и , если вариант уступает или не лучше варианта хотя бы по одному критерию. Если в -м столбце таблицы А все элементы равны нулю, то соответствующие варианты относятся к Парето-оптимальным, а если присутствует хотя бы одна единица, то такие варианты исключаются из дальнейшего рассмотрения как неэффективные. [20]

Таблица 25 Численные значения критериев вариантов

|  |  |  |
| --- | --- | --- |
| Критерий  Вариант |  |  |
|  | 4 | 5 |
|  | 5 | 2 |
|  | 1 | 4 |
|  | 5 | 3 |

Таблица 26 Сравнение вариантов алгоритмов на Парето-оптимальность

|  |  |  |
| --- | --- | --- |
| Варианты |  |  |
|  | 0 | 0 |
|  | 0 | 0 |
| Парето-оптимальность варианта | Да | Да |

Анализ данных, приведенных в таблицах 25 и 26, показывает, что все варианты сравниваемых алгоритмов являются Парето-оптимальными и требуется проведение их дальнейшего сравнения.

Для вычисления коэффициентов важности локальных критериев применим метод базового критерия. Будем придерживаться следующего порядка действий:

1) разобьем все множество локальных критериев на группы важности, при этом самые малозначимые критерии относим к первой (базовой) группе.

2) составим уравнение нормировки локальных критериев, в котором сумма коэффициентов важности всех локальных критериев должна быть равна единице.

. (3.2)

Где - количество групп важности локальных критериев;

- количество локальных критериев, которые входят в состав -ой группы

- коэффициент, который показывает степень превосходства критериев -ой группы по сравнению с критериями базовой группы, при этом

- коэффициент важности критериев первой группы, при

3) определим коэффициент важности () критериев базовой группы

(3.3)

4) определим коэффициенты важности () локальных критериев, входящих в состав -ой группы

(3.4)

5) выполним проверку условия нормировки локальных критериев

(3.5)

Следует иметь в виду, что

(3.6)

Где - количество локальных критериев, по которым проводим сравнение альтернативных вариантов алгоритмов.

Наиболее часто используют интегральный критерий взвешенной суммы для выбора наилучшего варианта из набора альтернативных вариантов алгоритмов. При этом локальными критериями являются показатели, по которым производят сравнение.

(3.7)

где — число показателей сравнения; — количество вариантов сравнения; cравнение вариантов алгоритмов ( ) осуществляется по параметрам ();

— коэффициент важности -го параметра сравнения. При этом обязательно должно быть выполнено условие нормировки критериев;

— коэффициент нормализации, определяет уровень соответствия -го параметра -го варианта наилучшему значению, .

Наилучшим вариантом алгоритма является вариант .

При нормализации критериев типа «чем больше, тем лучше» (например, производительность, гарантийный срок работы и т. д.) коэффициент нормализации находят согласно выражению

(3.8)

При нормализации критериев типа «чем больше, тем хуже» (например, габариты, масса, стоимость и т. д.) коэффициент нормализации определяют из выражения

(3.9)

где — значение -го локального критерия, соответствующее максимальному значению среди сравниваемых вариантов решения;

— значение -го локального критерия, соответствующее минимальному значению среди сравниваемых вариантов решения. [20]

Разобьем все показатели систем на группы важности.

В первую группу включаем два показателя (с кодовыми обозначениями , которые считаем самым малозначимым из набора показателей.

Во вторую группу включаем два показателя (с кодовыми обозначениями , который считаем наиболее значимыми, по сравнению с первыми в два раза.

Поэтому имеем следующие исходные данные:

- количество групп показателей сравнения систем;

- количество показателей, которые соответственно входят в состав 1-ой и 2-ой групп.

- коэффициенты, которые соответственно показывают степень превосходства критериев 2-ой над критериями 1-ой группы.

После подстановки данных в выражение 3.2 имеем:

Решая это уравнение, получаем

Подставляем вычисленное выражение ( в выражение 3.4 и получаем значения коэффициентов важности показателей сравнения систем:

Рассчитанные значения коэффициентов важности показателей сравнения алгоритмов, а также нормализованные значения этих показателей, вычисленные по формулам 3.8 и 3.9, приведены в таблице 27.

Таблица 27 Значения интегрального критерия

|  |  |  |  |
| --- | --- | --- | --- |
| Код локального критерия | Коэффициент важности локального критерия ( | Нормированное значение локального критерия | |
|  |  |
|  | 0,4 | 0,8 | 1 |
|  | 0,1 | 1 | 0,4 |
|  | 0,4 | 0,25 | 0,8 |
|  | 0,1 | 1 | 0,6 |
|  | | 0,62 | 0,82 |

Значения интегрального критерия приведены в таблице 27 (в последней строке) и представляют собой аддитивную свертку локальных критериев. Ранжирование вариантов алгоритмов показывает, что алгоритмы следует расположить в следующем порядке .При этом вариант алгоритма с кодовым названием является наилучшим среди сравниваемых.

В итоге получили, что алгоритм случайных разбиений подходит для разрабатываемой системы больше.

# Организационно-экономическая часть. Расчет затрат на создание программного изделия

## Смета затрат на программное изделие

Решение ряда экономических задач сопровождает процесс разработки автоматизированной системы. Расчёт стоимости информационно-программного изделия является важной задачей. Смета затрат включает следующие статьи:

1. расходные материалы;
2. затраты на оборудование;
3. затраты на оплату труда;
4. обобщенный социальный налог;
5. накладные расходы;
6. услуги сторонних организаций;
7. прибыль.

Таблица 28 Смета затрат на разработку автоматизированной системы

| № | Статья затрат | Сумма статьи (руб.) |
| --- | --- | --- |
| 1 | Расходные материалы | 1400 |
| 2 | Затраты на оборудование | 25500 |
| 3 | Услуги сторонних организаций | 2200 |
| 4 | Заработная плата |  |
| 5 | Страховые выплаты |  |
| 6 | Накладные расходы |  |
| 7 | Себестоимость |  |
| 8 | Цена (без НДС) |  |
| 9 | Чистая прибыль (после налогообложения) |  |

## Обоснование сметы затрат

### Расчёт затрат на расходные материалы

Затраты на расходные материалы приведены в табл. 29.

Таблица 29 Затраты на расходные материалы

|  |  |  |
| --- | --- | --- |
| Наименование | Количество | Цена (руб.) |
| Бумага формата А4 (высокого качества: плотность 80 г/м2). | 5 пачек (по 20 листов) | 200 |
| Бумага формата А1. | 8 листов | 50 |

|  |  |  |
| --- | --- | --- |
|  |  | (4.1) |

### Расчет затрат на оборудование

При разработке автоматизированной системы затраты по данной статье включают в себя расходы на использующуюся в разработке технику и её ремонт. Используемое оборудование представлено в табл. 30.

Таблица 30 Затраты на оборудование

|  |  |
| --- | --- |
| Наименование | Цена (руб.) |
| Ноутбук Asus X200L | 20000 |
| Сервер Cisco UCSC-C22-M3S-CH2 | 100000 |

|  |  |  |
| --- | --- | --- |
|  |  | (.2) |

Длительность использования оборудования составляет 9 месяцев. Затраты на вычислительную технику вычисляются по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (4.3) |

где – коэффициент амортизации на год для ускоренной амортизации, – стоимость компонента, – число компонентов, – период использования оборудования в месяцах. [21]

Таким образом, затраты на вычислительную технику при ускоренных сроках амортизации составляют

|  |  |  |
| --- | --- | --- |
|  |  | (4.5) |

Затраты на ремонт вычислительной техники составляют 10% от стоимости:

|  |  |  |
| --- | --- | --- |
|  |  | (4.6) |

Вычислим затраты на оборудование с учетом его ремонта:

|  |  |  |
| --- | --- | --- |
|  |  | () |

### Расчёт затрат на услуги сторонних организаций

Учтем также затраты на выполнение некоторых работ сторонними организациями работ. В табл. 31 приведены затраты на услуги сторонних организаций.

Таблица 31 Затраты на услуги сторонних организаций

|  |  |
| --- | --- |
| Наименование | Цена (руб.) |
| Вывод графической части на принтере | 1000 |
| Вывод графической части на плоттере | 200 |
| Переплет программной документации | 1000 |

|  |  |  |
| --- | --- | --- |
|  |  | (4.7) |

### Расчёт заработной платы

Рассчитаем заработную плату исполнителей, непосредственно связанных с разработкой ИПО, с учетом их должностного оклада и времени участия в разработке по следующей формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (4.8) |

где – основная заработная плата, – дополнительная заработная плата.

Затраты на выплату исполнителям заработной платы вычислим по следующему соотношению:

|  |  |  |
| --- | --- | --- |
|  |  | (4.9) |

где – количество разработчиков программного изделия, – минимальная заработная плата, – коэффициент, соответствующий разряду разработчика, – время разработки (в месяцах). [4, 21] Таким образом,

|  |  |  |
| --- | --- | --- |
|  |  | (4.10) |

Доходы физических лиц подлежат обязательному налогообложению (налог на доходы физических лиц), поэтому для компенсации выплат размер месячного оклада увеличивается, что отражено в следующей формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (4.11) |

где – налог на доходы физических лиц.

### Расходы на дополнительную заработную плату

Расходы на дополнительную заработанную плату включают все выплаты непосредственным исполнителям за время, не проработанное на производстве, но предусмотренное законодательством, в том числе: оплата отпусков, компенсация за недоиспользованный отпуск, и т.д. [4, 21] Величина этих выплат составляет 20% от размера основной заработной платы:

|  |  |  |
| --- | --- | --- |
|  |  | (4.12) |

В результате, получаем

|  |  |  |
| --- | --- | --- |
|  |  | (4.13) |

### Расчет отчислений на социальные нужды

Выплаты в пенсионный фонд, фонд социального страхования, фонд обязательного медицинского страхования, составляют, соответственно, 26%, 2,9%, 5,1%. [5] Таким образом отчисления на страховые взносы составят:

|  |  |  |
| --- | --- | --- |
|  |  | (4.14) |

### Расчет накладных расходов

Рассчитаем затраты на внепроизводственные расходы и расходы на управление. Накладные расходы составляют 10% от суммы остальных расходов:

|  |  |  |
| --- | --- | --- |
|  |  | (4.15) |

### Расчет прочих расходов

Прочими расходами является налог на имущество и налог на транспортные средства. [21]

Налог на имущество при разработке автоматизированной системы в рамках выпускной работы не платится, поскольку все имеющееся в наличии имущество, используется на нужды образования, и, следовательно, налогом на имущество не облагается.

Налог на владельцев транспортных средств не платится, в связи с отсутствием транспортных средств.

### Расчет себестоимости ИПО

Себестоимость ИПО рассчитываем как сумму всех вышеперечисленных затрат:

|  |  |  |
| --- | --- | --- |
|  |  | (4.16) |

### Расчет цены

Рассчитаем цену без НДС (поскольку программный продукт не облагается НДС согласно статье 149 п. 26 Налогового кодекса РФ от 05.08.2000 N 117-ФЗ (действующая редакция от 06.04.2015)), добавив 20% стоимости:

### Расчёт чистой прибыли до и после налогообложения

В качестве схемы налогообложения была выбрана схема упрощенного налогообложения «доходы минус расходы», для которой отчисления составляют 15% от чистой прибыли. Чистая прибыль до налогообложения составляет

|  |  |  |
| --- | --- | --- |
|  |  | (4.17) |

После налогообложения чистая прибыль составляет

|  |  |  |
| --- | --- | --- |
|  |  | (4.18) |

# Заключение

В результате выполнения квалификационной работы были получены следующие результаты:

1. изучена предметная область и проведены обоснование и выбор технологий для создания системы на основе предъявленных к ней требований;
2. разработана архитектура системы;
3. разработана модель хранения данных и структура базы данных системы;
4. исследованы алгоритмы для разбиения числа на различные группы цифр;
5. разработан и реализован интерфейс пользователя для клиентской части автоматизированной системы;
6. проведено организационно-экономическое обоснование разработки;
7. разработана техническая документация к системе;
8. разработана графическая часть конструкторской документации.

# Список использованных источников

|  |  |
| --- | --- |
| 1. | ГОСТ 19.201-78. ЕСПД. Техническое задание. Требования к содержанию и оформлению. |
| 2. | ГОСТ 19.301-79. ЕСПД. Программа и методика испытаний. Требования к содержанию и оформлению. |
| 3. | ГОСТ 19.404-79. ЕСПД. Пояснительная записка. Требования к содержанию и оформлению. |
| 4. | Трудовой кодекс Российской Федерации от 30.12.2001 N 197-ФЗ (ред. от 03.12.2012) // «Российская газета», N 256, 31.12.2001 |
| 5. | Федеральный закон от 24.07.2009 N 212-ФЗ (ред. от 25.12.2012) «О страховых взносах в Пенсионный фонд Российской Федерации, Фонд социального страхования Российской Федерации, Федеральный фонд обязательного медицинского страхования» // «Собрание законодательства РФ», 27.07.2009, N 30, ст. 3738 |
| 6. | Айвалиотис Д. А. Администрирование сервера NGINX, - М.: МК Пресс, 2012. |
| 7. | Баррет Д. Дж. Linux: основные команды. Карманный справочник, - М.: КУДИЦ-ОБРАЗ, 2008. |
| 8. | Гифт Ноа, М. Джонс Джереми Python в системном администрировании UNIX и Linux, - М.: Символ-Плюс, 2009. |
| 9. | Головатый А., Каплан-Мосс Д. «Django. Подробное руководство», - М.: Символ-Плюс, 2011. |
| 10. | Лутц Марк Программирование на Python. Том 1, 4-е издание, - М.: Символ-Плюс, 2011. |
| 11. | Лутц Марк Программирование на Python. Том 2, 4-е издание, - М.: Символ-Плюс, 2011. |
| 12. | Форсье Джефф, Биссекс Пол Django. Разработка веб-приложений на Python, - М.: Символ-Плюс, 2011. |
| 13. | Документация nginx URL: <http://nginx.org/ru/> (дата обращения: 05.05.2016). |
| 14. | Интернет-справочник по языку HTML. URL: <http://www.htmlbook.ru> (дата обращения: 11.05.2016). |
| 15. | Руководство по внедрению и использованию Twitter Bootstrap URL: <http://getbootstrap.com/> (дата обращения: 05.05.2016) |
| 16. | Сайт Django Project. URL: <https://www.djangoproject.com> (дата обращения: 11.05.2016) |
| 17. | Документация PostgreSQL 9.3. URL: www.postgresql.org (дата обращения: 10.05.2016). |
| 18. | Документация Redis. URL: www.redis.io (дата обращения: 11.05.2016) |
| 19. | Документация uWSGI. URL: www.uwsgi-docs.readthedocs.io (дата обращения 12.05.2016) |
| 20. | Постников В. М., Черненький В. М. Методы принятия решений в системах организационного управления. М.: Издательство МГТУ им. Баумана, 2014. |
| 21. | Скворцов Ю. В., Некрасов Л. А. Организация и планирование машиностроительного производства. М.: Высшая школа, 2003. |
| 22. | Проект о мнемонике // Главная страница сайта powermemory.ru URL: www.powermemory.ru (дата обращения: 03.05.2016) |
| 23. | Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ / Пер. с англ. под ред. А. Шеня. – М.: МЦНМО, 2002. – 960 с.: 263 ил. |
| 24. | Портал аналитической информации, справочные материалы по разработкам баз данных (разработка инфологических и даталогических моделей). URL: <http://www.citforum.ru> (дата обращения: 15.05.2016) |
| 25. | Оптимальный способ запоминания цифр // Статья на проекте Habrahabr. URL: https://habrahabr.ru/post/141475/ (дата обращения: 28.04.2016) |
| 26. | Виленкин Н. Я. Популярная комбинаторика. М: Наука, 1975. |