## АННОТАЦИЯ

Отчет о курсовой работе: \*\*\* с., \*\*\* рис., \*\*\* табл., \*\*\* приложение, \*\*\* источников.  
 Объект исследования – матрицы и операции над ними.

Цель работы – разработка веб-сайта для произведения математических операций над матрицами.

Метод исследования – изучение принципов разработки веб-приложений средствами фреймворка Django.

В работе были использованы технологии: HTML, CSS, Java Script, Pyhton, Django, PyCharm, AJAX.

В результате решения задачи был разработан сайт-калькулятор для произведения математических операций над матрицами разной размерности.

WEB-ПРИЛОЖЕНИE, JQUERY, HTML, CSS, JAVA SCRIPT, PYHTON, DJANGO, PYCHARM, AJAX

|  |  |  |
| --- | --- | --- |
| СОДЕРЖАНИЕ | | |
|  | ВВЕДЕНИЕ | 5 |
| 1 | Анализ предметной области | 6 |
|  | 1.1 Состояние вопроса | 6 |
|  | 1.2 Актуальность и цель работы | 8 |
| 2 | Техническое задание | 9 |
|  | 2.1 Общие требования к продукту | 9 |
|  | 2.2 Позиционирование продукта | 10 |
|  | 2.3 Функции продукта | 21 |
|  | 2.4 Сценарии использования продукта | 22 |
|  | 2.5 Функциональные требования | 26 |
| 3 | Проектирование программного продукта | 30 |
|  | 3.1 Обоснование средств разработки | 30 |
|  | 3.2 Проектирование математической модели |  |
|  | \*\*\* Проектирование взаимодействия клиентской и серверной части |  |
| 4 | Разработка ПО | 41 |
|  | 4.1 Разработка пользовательского интерфейса |  |
|  | 4.2 Разработка серверной части | 41 |
|  | 4.3 Разработка взаимодействия клиентской и серверной части | 49 |
| 5 | Описание программного продукта | 56 |
|  | 5.1 Структура проекта | 56 |
|  | 5.2 Описание объектов и их взаимодействия | 58 |
|  | 5.3 Описание запросов | 60 |
| **6** | **Тестирование и внедрение** | **62** |
|  | 6.1 Тестирование разработанного ПО | **62** |
|  | **6.2** Установка программы | **64** |
| **ЗАКЛЮЧЕНИЕ** | | **68** |
| **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** | | **69** |
| **ПРИЛОЖЕНИЕ А (обязательное) Фрагменты листинга** | | **70** |
| **ПРИЛОЖЕНИЕ Б (обязательное) Экранные формы** | |  |
|  | |  |
|  | |  |
|  | |  |

## ВВЕДЕНИЕ

Термин «матрица» имеет много значений. Основное значение термин «матрица» имеет в математике. Ма́трица — математический объект, записываемый в виде прямоугольной таблицы элементов кольца или поля (например, целых или комплексных чисел), которая представляет собой совокупность строк и столбцов, на пересечении которых находятся её элементы. Количество строк и столбцов матрицы задают размер матрицы. Хотя исторически рассматривались треугольные матрицы, в настоящее время говорят исключительно о матрицах прямоугольной формы, так как они являются наиболее удобными и общими. Матрицы широко применяются в математике для компактной записи систем линейных алгебраических или дифференциальных уравнений. В этом случае, количество строк матрицы соответствует числу уравнений, а количество столбцов — количеству неизвестных. В результате решение систем линейных уравнений сводится к операциям над матрицами. Поэтому, целью курсовой работы стало создание веб-приложение для произведения операций над матрицами.

Исходя из указанной цели, можно выделить частные задачи, поставленные в курсовой работе:

1. Выделить основные операции над матрицами.
2. Проанализировать существующие аналоги и их реализацию.
3. В качестве результата анализа составить ТЗ для веб-приложения.
4. Изучить технологии: HTML, CSS, Java Script, Pyhton, Django, PyCharm, AJAX.
5. Разработать структуру и функционал веб-приложения, соответствующий техническому заданию.
6. Разработать клиентскую и серверную части программы.
7. Проанализировать разработанное веб-приложение.

**1 Анализ предметной области**

* 1. Состояние вопроса

Впервые матрицы упоминались ещё в древнем Китае, называясь тогда «волшебным квадратом». Основным применением матриц было решение линейных уравнений. Так же, волшебные квадраты были известны чуть позднее у арабских математиков, примерно тогда появился принцип сложения матриц. После развития теории определителей в конце 17-го века, Габриэль Крамер начал разрабатывать свою теорию в 18-ом столетии и опубликовал «правило Крамера» в 1751 году. Примерно в этом же промежутке времени появился «метод Гаусса». Теория матриц начала своё существование в середине XIX века в работах Уильяма Гамильтона и Артура Кэли. Фундаментальные результаты в теории матриц принадлежат Вейерштрассу, Жордану, Фробениусу. Термин «матрица» ввел Джеймс Сильвестр в 1850 г.

С развитием технического прогресса теория матриц находит все новые применения. Например, в программировании матрица – это двумерный массив, в электронике – набор проводников, которые можно замкнуть в точках их пересечений и так далее. Также, актуальность решения операций над матрицами возникает в сфере образования для проверки рений.

С целью выявления недостатков и достоинств, уточнения терминологии, были проанализированные некоторые из существующих аналогов: «Образовательные веб-сервисы» [\*\*\*], «Онлайн-калькулятор Math» [\*\*\*]. Данные аналоги были выбраны из общей массы, по причине того, что они являются одними из самых рейтинговыми в поисковых сетях Google и Яндекс.

|  |  |
| --- | --- |
| Критерий | Работа с матрицами |
| Название | Образовательные веб-сервисы |
| Страна | Россия |
| Функционал | На сайте собраны калькуляторы для решения матриц онлайн: нахождение определителя, умножение и сложение матриц, нахождение обратной матрицы и многие другие. |
| Достоинства | 1. У каждой программы есть описание, информация о данном действии над матрицей и формула, используемая программой для решения. 2. Онлайн калькуляторы подробно расписывают ход решения, что позволяет не только получить результат, но и научиться решать матричные задачи, найти и исправить ошибки или проверить правильность своего решения. |
| Недостатки | 1. Функционал данного сервиса разбит на отдельные веб-приложения. 2. При каждом вводе данных (выбор ранга и ячеек матрицы) веб-приложение перенаправляет пользователя на отдельную веб-страницу. |

|  |  |
| --- | --- |
| Критерий | Работа с матрицами |
| Название | Онлайн-калькулятор Math |
| Страна | Россия |
| Функционал | Используется для проверки своего решения по многим математическим и экономическим дисциплинам. |
| Достоинства | 1. У каждой программы есть описание, информация о данном действии над матрицей и формула, используемая программой для решения. 2. Результат решения - это отчет в формате Word (и Excel при необходимости), содержащий ход решения с комментариями, исходные формулы и выводы. |
| Недостатки | 1. Функционал данного сервиса разбит на отдельные веб-приложения. 2. После ввода данных происходит перезагрузка страницы. 3. Если в браузере включена блокировка рекламы, то решение появляется только через 30 секунд. |

В ходе анализа были обнаружены недостатки систем, связанные с тем, что для каждой операции существует отдельный сервис, а также с повышенным трафиком на интернет, так как происходит перезагрузка страницы.

1.2 Актуальность и цель работы

Создание разрабатываемого веб-приложения, актуально, так как актуальность произведения операций над матрицами растет, а текущие аналоги обладают не всеми нужными характеристиками для удовлетворения нужд пользователей. Внедрение разрабатываемого веб-приложения позволит:

1. Создавать квадратные и прямоугольные матрицы произвольных рангов
2. Производить основные операции над матрицами

Целью курсовой работы является разработка веб-приложения, которое позволит производить основные математические операции над матрицами.

Исходя из указанной цели, можно выделить частные задачи, поставленные в курсовой работе:

1. Выделить основные операции над матрицами.
2. Проанализировать существующие аналоги и их реализацию.
3. В качестве результата анализа составить ТЗ для веб-приложения.
4. Изучить технологии: HTML, CSS, Java Script, Pyhton, Django, PyCharm, AJAX.
5. Разработать структуру и функционал веб-приложения, соответствующий техническому заданию.
6. Разработать клиентскую и серверную части программы.
7. Проанализировать разработанное веб-приложение.

**2 Техническое задание**

2.1 Общие требования к продукту

Основная задача веб-приложения – произведение основных математических операций с матрицами. Из анализа существующих аналогов можно выделить следующие требуемые особенности веб-приложения:

1. Все операции и ввод данных должны производится в рамках одной веб-страницы
2. Результаты вычислений должны выводится на той же веб-странице без перезагрузки
3. В силу особенностей некоторых операций, следует ограничить пользователя при введении рангов матриц и данных

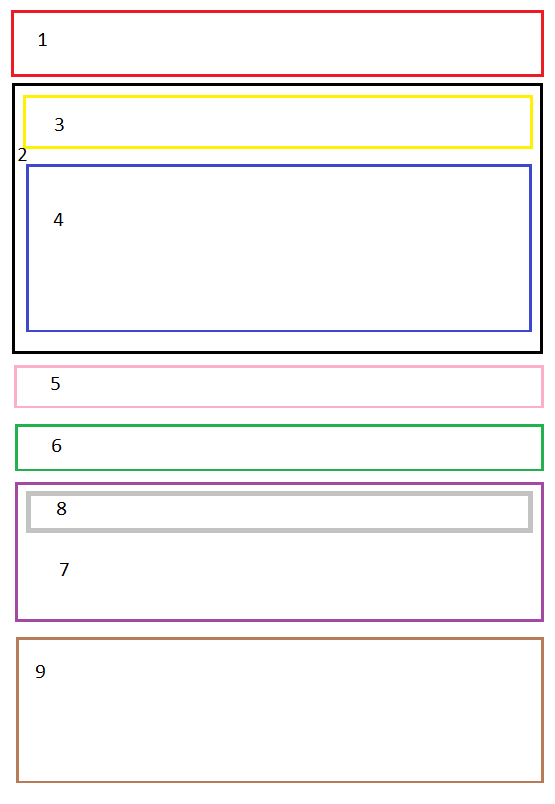
Интерфейс программы должен быть простым и понятным, неиспользуемые поля для некоторых операций следует скрыть, а нужные отображать с подсказкой об их функциональности.

2.2 Позиционирование продукта

2.2.1 Требование к пользовательским интерфейсам

Интерфейсом веб-приложения является веб-страница, отображаемая при входе на сайт. При ее создании в дизайне следует придерживаться простоты и понятности, а также текущих стандартов интерфейса веб-страниц.

Элементы веб-приложения следует разместить в вертикальной последовательности с подписями к каждому из них, как показано на рисунке \*\*\*.



На рисунке \*\*\* показаны следующие элементы интерфейса:

1. «Шапка» сайта с названием веб-приложения и кратким описанием
2. Блок для ввода начальной матрицы
3. Поля для ввода размеров матрицы
4. Таблица, содержащая поля для ввода значений ячеек матрицы
5. Элемент для выбора операции из списка
6. Поле для ввода числа (для операции «Умножение матрицы на число»)
7. Блок для ввода матрицы-операнда (для операций сложения, вычитания и умножения матриц)
8. Блок с элементом для ввода ширины матрицы-операнда (для операции «Умножение матриц»)
9. Блок с таблицей результирующей матрицы.

Для всех блоков, которые предназначены для некоторых операций реализовать их автоматическое появление и скрытие при выборе операций в соответствующем блоке. Вывод результата операции «Определитель матрицы» выводить в качестве модального окна.

2.2.2 Требования к программным интерфейсам

Код программы должен быть понятным, содержать комментарии, не содержать повторяющегося кода и быть готовым для дальнейшей разработки. Архитектура веб-приложения должна быть реализована в соответствии со стандартами MVC, а операции с матрицами должны быть реализованы в качестве отдельного python-модуля.

* + 1. Требования к пользователям продукта
* Подключение к сети Интернет.
* Базовый навык работы с компьютером.

2.3 Функции продукта

Требуемый функционал продукта:

* Создание матрицы произвольного размера
* Заполнение созданной матрицы
* Выполнение основных математических операций над матрицами

2.4 Сценарии использования продукта

Начальный сценарий: пользователь открывает веб-приложение в браузере.

Сценарии выбора математической операции:

Сценарий 1: Пользователь выбирает операцию сумма/вычитание матриц. На веб-странице ниже появляется матрица, такого же размера, как и исходная матрица.

Сценарий 2: Пользователь выбирает операцию умножение матриц. На веб-странице ниже появляется матрица, такой же высоты, как и у исходной матрицы, а над ней появляется элемент для выбора ширины матрицы-операнда.

Сценарий 3: Пользователь выбирает операцию умножение матрицы на число. На веб-странице ниже появляется блок для ввода числа, при этом блок для ввода второй матрицы пропадает

Сценарий 4: Пользователь выбирает операцию нахождения определителя или транспонирования матрицы. Все элементы для ввода вторых операндов пропадают.

Сценарии при нажатии кнопки подсчета:

Сценарий 1: Выбрана операция нахождения определителя. В результате появляется модальное окно с найденным определителем.

Сценарий 2: Выбрана операция отличная от нахождения определителя. В результате ответ помещается в блок с результирующей таблицей.

Сценарий 3: Выбрана любая операция, но выбраны не корректные данные. Появляется модальное окно, оповещающее об ошибке.

**3. Реализация программного продукта**

3.1 Обоснование средств разработки

3.1.1 HTML5

HTML5 (англ. HyperText Markup Language, version 5) — язык для структурирования и представления содержимого всемирной паутины. Это пятая версия HTML. Хотя стандарт был завершён (рекомендованная версия к использованию) только в 2014 году (предыдущая, четвёртая, версия опубликована в 1999 году), ещё с 2013 года браузерами оперативно осуществлялась поддержка, а разработчиками — использование рабочего стандарта (англ. HTML Living Standard). Цель разработки HTML5-улучшение уровня поддержки мультимедиа-технологий с одновременным сохранением обратной совместимости, удобочитаемости кода для человека и простоты анализа для парсеров.

В HTML5 реализовано множество новых синтаксических особенностей. Например, элементы <video>, <audio> и <canvas>, а также возможность использования SVG и математических формул. Эти новшества разработаны для упрощения создания и управления графическими и мультимедийными объектами в сети без необходимости использования сторонних API и плагинов. Другие новые элементы, такие как <section>, <article>, <header> и <nav>, разработаны для того, чтобы обогащать семантическое содержимое документа (страницы). Новые атрибуты были введены с той же целью, хотя ряд элементов и атрибутов был удалён. Некоторые элементы, например, <a>, <menu> и <cite>, были изменены, переопределены или стандартизированы.

* + 1. CSS3

CSS3 (англ. Cascading Style Sheets 3 — каскадные таблицы стилей третьего поколения) — активно разрабатываемая спецификация CSS. Представляет собой формальный язык, реализованный с помощью языка разметки. Самая масштабная редакция по сравнению с CSS1, CSS2 и CSS2.1. Главной особенностью CSS3 является возможность создавать анимированные элементы без использования JS, поддержка линейных и радиальных градиентов, теней, сглаживания и многое другое.

CSS3 преимущественно используется как средство описания и оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL.

В отличие от предыдущих версий спецификация разбита на модули, разработка и развитие которых идёт независимо. CSS3 основан на CSS2.1, дополняет существующие свойства и значения и добавляет новые нововведения, начиная с малых, вроде закругленных углов блоков, заканчивая трансформацией (анимацией) и, возможно, введением переменных.

3.1.3 JavaScript

JavaScript — прототипно-ориентированный сценарный язык программирования. Является реализацией языка ECMAScript (стандарт ECMA-262).

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java, но при этом лёгким для использования непрограммистами. Языком JavaScript не владеет какая-либо компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке.

3.1.4 Twitter Bootstrap

Пользовательским интерфейсом является всплывающая, при нажатии на иконку расширения, HTML-страница. Для создания данной страницы использовался фреймворк Bootstrap 3 [\*\*\*]. Он предоставляет набор интрументов для создания сайтов и различных веб-страниц. Данный фреймворк включает в себя HTML и CSS шаблоны, а также включает в себя расширения Javascript.

Bootstrap позволяет создать адаптивный сайт с помощью резиновой системы разметки, которая масштабируется до 12 столбцов на различных устройсвах. Подобный масштаб позволяет создавать как простые варианты разметки, так и более сложные макеты.

3.1.5 Django framework

Django (Джанго) — свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других (например, Ruby on Rails). Один из основных принципов фреймворка — DRY (англ. Don't repeat yourself)

Также, в отличие от других фреймворков, обработчики URL в Django конфигурируются явно при помощи регулярных выражений, а не выводятся автоматически из структуры моделей контроллеров.

Для работы с базой данных Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных.

3.1.6 PyCharm IDE

PyCharm — интегрированная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов. PyCharm разработана компанией JetBrains на основе IntelliJ IDEA.

Основным критерием выбора данной IDE является то, что она поддерживает веб-разработку на Django. Среди возможностей PyCharm, помогающих в разработке проекта можно выделить следующие:

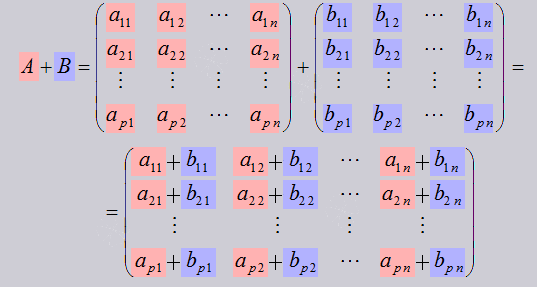
* Создание каркаса проекта Django
* Запуск тестового сервера Django, нажатием одной кнопки
* Отображение лога сервера в интегрированной в IDE консоли

3.2 Проектирование математической модели

Матрицей называется прямоугольная таблица из чисел, содержащая некоторое количество m строк и некоторое количество n столбцов. Числа m и n называются порядками матрицы. В случае, если m = n, матрица называется квадратной, а число m = n - ее порядком.

Основными арифметическими операциями над матрицами являются умножение матрицы на число, сложение и умножение матриц.

Операция сложения определена только для матриц одного порядка. Сумма двух матриц А и B - это матрица, элементы которой равны сумме соответствующих элементов матриц А и В, как показано на рисунке \*\*\*.

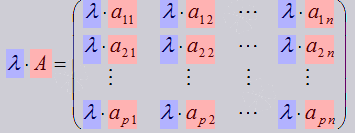


Из определения суммы матриц вытекает, что операция сложения матриц обладает теми же свойствами, что и операция сложения вещественных чисел, а именно:

1. Для матриц А, В и С одного порядка характерно свойство ассоциативности сложения А+(В+С) = (А+В)+С.
2. Для матриц данного порядка существует нейтральный элемент по сложению, которым является нулевая матрица. То есть, справедливо свойство А+О=А.
3. Для ненулевой матрицы А данного порядка существует матрица (–А), их суммой является нулевая матрица: А+(-А) = О.
4. Для матриц А и В данного порядка справедливо свойство коммутативности сложения А+В=В+А.

Операцию вычитания можно вывести из третьего свойства операции сложения (A-B = A+(-B), т.к. B = - (-B)).

Произведение матрицы и действительного (или комплексного) числа - это матрица, элементы которой получаются умножением соответствующих элементов исходной матрицы на число (рисунок \*\*\*). Операция умножения матрицы на число определена для матриц любого порядка.

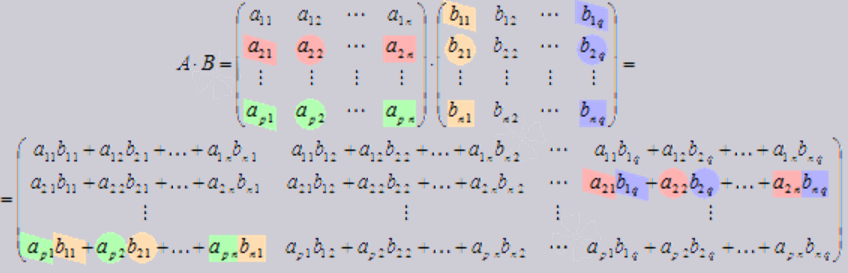


Свойства операции умножения матрицы на число:

1. Для матриц одного порядка А и В, а также произвольного действительного (или комплексного) числа справедливо свойство дистрибутивности умножения относительно сложения.
2. Для произвольной матрицы А и любых действительных (или комплексных) чисел выполняется свойство дистрибутивности.
3. Для произвольной матрицы А и любых действительных (или комплексных) чисел справедливо свойство ассоциативности умножения.
4. Нейтральным числом по умножению на произвольную матрицу А является единица.

Из свойств операции умножения матрицы на число следует, что умножение нулевой матрицы на число ноль даст нулевую матрицу, а произведение произвольного числа и нулевой матрицы есть нулевая матрица.

Произведение матрицы А порядка m\*n и матрицы В порядка n\*k - это такая матрица С m\*k, каждый элемент которой равен сумме произведений элементов i-ой строки матрицы А на соответствующие элементы j-ого столбца матрицы В, как на рисунке \*\*\*. Операция умножения двух матриц, A и B определяется только для случая, когда число столбцов матрицы A равно числу строк матрицы B.

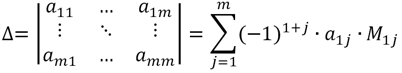


Если матрицы А, В и С подходящих порядков, то справедливы следующие свойства операции умножения матриц.

1. Свойство ассоциативности умножения матриц A\*(B\*C) = (A\*B)\*C.
2. Два свойства дистрибутивности (A+B)\*C = A\*C + B\*C и A\*(B+C) = A\*B+A\*C.
3. В общем случае операция умножения матриц некоммутативна.

Определитель (детерминант) квадратной матрицы — это число, которое ставится в соответствие матрице и вычисляется по ее элементам согласно следующим правилам:

* Для матрицы первого порядка значение определителя равно единственному элементу этой матрицы
* Для матриц более высоких порядков (выше второго) m×m определитель можно вычислить, применив рекурсивную формулу, как это показанно на рисунке \*\*\*, где Mij - дополнительный минор квадратной матрицы — определитель матрицы, полученной из исходной вычеркиванием i-ой строки и j-ого столбца.



**4. Разработка ПО**

4.1 Создание пользовательского интерфейса

4.1.1 Верстка веб-страницы

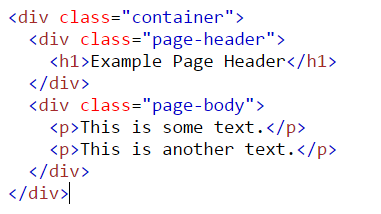
Интерфейсом разработываемого ПО является веб-страница, которая отображается браузером при заходе на сайт. Для создания веб-страницы сайта использовался фреймворк Twitter Bootstrap. Он предоставляет набор интрументов для создания сайтов и различных веб-страниц. Данный фреймворк включает в себя HTML и CSS шаблоны, а также включает в себя расширения Javascript.

Bootstrap позволяет создать адаптивные веб-страницы с помощью резиновой системы разметки, которая масштабируется до 12 столбцов на различных устройсвах. Подобный масштаб позволяет создавать как простые варианты разметки, так и более сложные макеты. Для верстки сайта использовались классы, описанные в таблице \*\*\*.

Таблица \*\*\* – Используемые css-классы при верстке

|  |  |
| --- | --- |
| Класс | Описание |
| Сontainer | Блок-обертка всего содержимого, задает основную ширину сайта |
| Row | Базовый блок для колонок |
| Modal | Класс для создания модального окна |
| Nav-tabs | Класс для создания переключателей вкладок |

В таблице \*\*\* отображены основные классы, которые неоднократно использовались в проекте. По мимо данных, использовались и дополнительные классы, которые помогают позиционировать текст в блоках, прятать блоки, выделять текст особым цветом и так далее. Также, основные классы Bootstrap3 используют вложенные подклассы для разделения содержимого блоков. Так, например, класс Modal использует подкласс modal-dialog, в котором должен содержаться блок с классом modal-content, который содержит три блока с классами: modal-header, modal-body, modal-footer. Похожую иерархию имеют и блоки с классом container, пример реализации которой можно подробнее рассмотреть на рисунке \*\*\*.



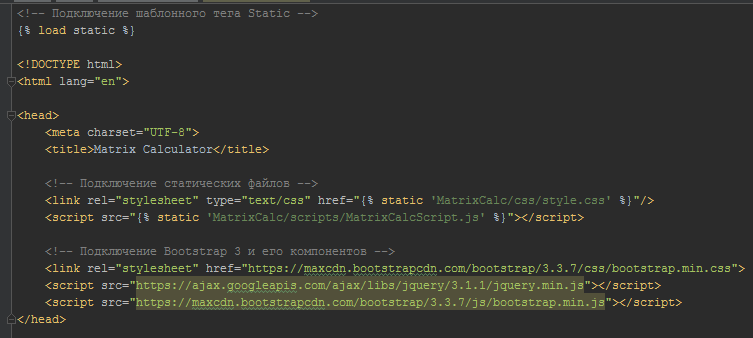
Система разметки Bootstrap также имеет собственные правила размещения элементов, которые рассмотрим ниже:

1. Элементы row всегда помещаются в container (с фиксированной шириной) или в container-fluid (полная ширина) для корректного выравнивания и наполнения.
2. Строки используются для создания горизонтальной группы столбцов.
3. В столбцах помещается контент, только колонки могут быть непосредственными потомками строк.
4. Для быстрого создания сетки макетов используются стандартные классы сетки (например, row или col-xs-4).
5. Колонки имеют собственный внутренний отступ (padding). Для первой и последней колонки он убран с помощью отрицательного отступа margin.
6. Столбцы сетки планируются с учетом максимального ограничения – 12 столбцов. Следовательно, чтобы создать три блока, нужно использовать класс col-md-4
7. Если ограничение превышено – каждая следующая колонка переносится на новый ряд.

Однако, перед отправкой веб-страницы пользователю она компилируется в самом Django, что позволяет использовать «шаблонные теги и фильтры». Данные теги помогают сделать страницу более динамичной, так как есть возможность передать туда данные из контроллера. Существует много шаблонных тегов в Django, однако для разработки понадобятся следующие:

* Load – загрузка библиотеки тегов
* Csrf\_token - в Django 1.2 и выше, используется для CSRF защиты
* Static - создает ссылку на файл в директории со статическими файлами

Чтобы воспользоваться шаблонным тегом, необходимо указать его имя и его параметры через пробел между «{%» и «%}» прямо внутри разметки веб-страницы. Первым, используется тег load с параметром static, что позволит использовать последний в шаблоне. Шаблонный тег static используется для указания пути к статическим файлам в тегах link и script. Тег сsrf\_token доступен и без подключения, и для его использования данный шаблонный тег нужно исользовать в любом месте тега form. Таким образом, подключение файла с бедующим скриптом для страницы будет выглядеть, как на рисунке \*\*\*.



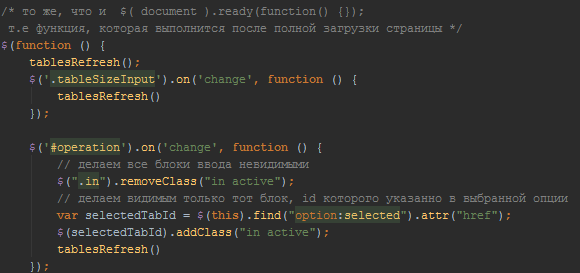
4.1.2 Разработка пользовательских скриптов

Для создания матрицы на стороне клиента, без перезагрузки веб-страницы, требуется либо постоянно отправлять AJAX запрос на сервер, либо генерировать таблицу с помощью Java Script. Второй вариант выглядит более оптимально, так как будет выполнятся быстрее и не будет занимать интернет-трафика. Для его реализации требуется создать функцию tableRefresh внутри подключённого к веб-странице скрипты, в которой будет происходить обновление веб-страницы. Далее, нужно, чтобы при изменении значения в тегах input, которые задают размеры матриц, срабатывала функция обновления таблиц. Это можно сделать разными способами, однако самым удобным будет следующий способ:

1. Ко всем полям ввода добавить специальный класс tableSizeInput
2. Использовать функцию on библиотеки jquery c параметрами «change» и экземпляром функции (которая обновляет таблицы) ко всем тегам с классом tableSizeInput.

При изменении операции некоторые блоки должны скрываться, а некоторые наоборот – становится видимыми, при этом размеры второй и результирующей таблицы тоже могут поменяется. Функцию по событию изменения элемента можно сделать средствами той же функции on библиотеки jquery, обновить таблицы с помощью функции tableRefresh. Все необходимые блоки, которые нужно скрыть содержат два класса: tab-pane и fade. Данные классы делают блок невидимым, пока к блоку не добавятся еще два класса: in и active. Данную возможность можно использовать, чтобы выборочно отображать блоки ввода.

Поскольку, скрипт подключается до тела страницы, то невозможно в скрипте добавить обработчики событий к элементам, так как они еще не существуют. Данную проблему можно решить, воспользовавшись той же функцией on к документу на событие загрузки. Также, можно использовать данное событие для запуска обновления таблиц, после чего скрипт при запуске будет выполнять код, изображенный на рисунке \*\*\*.



После осуществления динамичности содержимого страницы, следует решить задачу с отправкой и поучением данных на будущий сервер. Для реализации технологии AJAX существует много библиотек, однако, поскольку уже подключена библиотека jquery, то можно воспользоваться ее методом ajax, для отправки данных на сервер. Также, полезной будет функция serialize, которая превращает данные формы в json-обьект. Код, описывающий создание асинхронного запроса к серверу изображен на рисунке \*\*\*.



Функция fillAnswerTable заполняет результирующую матрицу данными, основываясь на таблице, присланной от сервера в формате json. Листинг данной и других вспомогательных функций можно будет изучить в листинге \*\*\*.

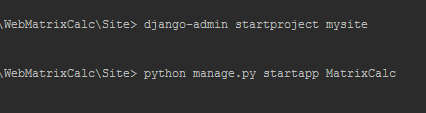
4.2 Разработка серверной части

4.2.1 Создание проекта Django

Процедура создания начального Django-проекта состоит из следующих шагов:

1. Установка интерпретатора Python
2. Установка библиотеки Django
3. Создание проекта через один из управляющих скриптов Django
4. Создание приложения внутри проекта

Для создания проекта, а затем приложения Django в OC Windows следует вызвать командную строку и там прописать команды, как на рисунке \*\*\*.



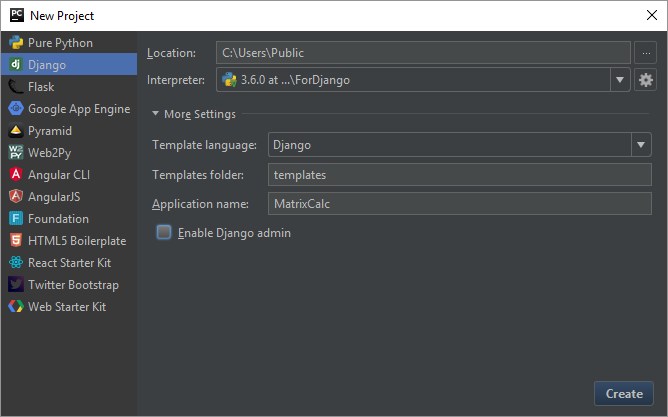
В первой команде на рисунке \*\*\* команда обращается к общему командному файлу Django для создания проекта, о чем говорит параметр startproject, а параметр mysite устанавливает имя проекта, которое влияет разве-что на имя папки, в которой будет содержатся проект.

После создания проекта его уже можно запустить, однако, по одному из основных принципов фреймворка Django «Don’t repeat yourself» следует создать еще приложение внутри проекта, что можно сделать из второй команды рисунка \*\*\*. В данном случае, выполняется метод startapp с переданным параметром названия приложения, который содержится в одном из модулей проекта. Данный модуль, также может запускать проект на локальном тестовом сервере и многое другое.

Использование PyCharm IDE при разработке ПО значительно сократить вышеупомянутый алгоритм. Для этого нужно вызвать окно диалога для создания нового проекта, затем выбрать тип проекта (Django). Далее следует выполнить следующие шаги:

* Выбрать каталог для создания проекта
* Выбрать интерпретатор Python (если на нем не будет установлен Django, то он автоматически установится)
* Выбрать название папки для хранения шаблонов
* Выбрать название приложения (если не будет введено, то проект будет создан без приложения)

После заполнения необходимых полей окно диалога будет выглядеть, как на рисунке \*\*\*.



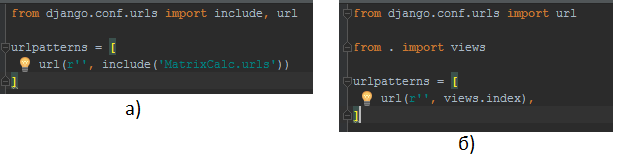
Также, в проект следует добавить файл Matrix.py, который в будущем будет содержать все методы по работе с матрицами. Его можно создать в любом месте проекта, однако, на данном этапе, можно создать его именно внутри приложения MatrixCalc.

4.2.2 Разработка контроллера

После создания проекта, его уже можно будет запустить на локальном сервере Django, однако, прежде, нужно настроить сервер на отображение уже созданного шаблона.

Первое, что нужно сделать после создания проекта – это настроить файлы urls.py. Этот модуль содержит код Python, который отображает URL-шаблоны (регулярные выражения) и связанные функции Python (ваши представления). Этот модуль может быть короткий или длинный настолько, насколько это нужно. Он может ссылаться на другие такие модули. И, так как это код Python, может создаваться динамически.

В нашем случае код менеджеров URL будут достаточно просты. Основной модуль, который находится в корне проекта предназначен, в основном, для переадресации URL-ов на другие модули приложений для дальнейшей обработки. Это сделано для того, чтобы не загромождать данный файл и не выносить логику обработки URL-ов приложений в файл корня проекта. Внутренний менеджер URL-адресов приложения связывает URL и контроллер, который его обрабатывает. На рисунке \*\*\* изображено содержание менеджеров URL-адресов.



Как можно заметить на рисунке \*\*\* (б) к пустому адресу URL привязан контроллер index, которого еще не существует, поэтому при попытке запуска проекта будет ошибка.

Контроллер можно создать в любом модуле проекта, однако для него изначально был создан пустой файл views в папке с приложением. Контроллером является обычная функция Python, принимающая один параметр – обьект запроса, который содержит все данные о запросе. Среди данных основным является метод запроса (GET или POST), что дает понять внутри контроллера, что следует делать – обрабатывать данные, присланные в запросе, либо просто вернуть веб-страницу в ответ на GET-запрос.

Для того, чтобы вернуть веб-страницу в контроллере – ее сперва следует создать из шаблона. Для этого в Django есть несколько функций, отличающихся некоторыми параметрами. Воспользуемся функцией render, которая выполняет указанный шаблон с переданным словарем контекста и возвращает HttpResponse с полученным содержимым.

Теперь, при обращении к нашему сайту в браузере будет отображаться веб-страница, которая была создана из ранее составленного шаблона. Однако, при отправке формы страница перезагружается и ничего не происходит. Это происходит потому, что форма отправляется по тому же URL, что и сам сайт, по этому его обрабатывает тот же контроллер, который может только возвратить страницу. Для того, чтобы исправить это, следует в зависимости от типа запроса (GET или POST) выдавать разные результаты. В случае GET-запроса возвращать веб-страницу, а в случае POST – возвращать данные json.

В целом, логику модуля при получении POST-запроса можно разбить на следующие составляющие:

1. Формирование матриц и создание необходимых переменных на основе данных, переданных в POST-запросе
2. Действие над полученными данными, в зависимости от переданной операции
3. Возвращение данных в формате json

Для второго шага использовались нереализованные методы библиотеки Matrix, которая будет разрабатываться в дальнейшем, а для последнего шага можно воспользоваться функцией JsonResponse, модуля http фреймворка Django. Данная функция преобразовывает переданный в нее словарь в json-строку.

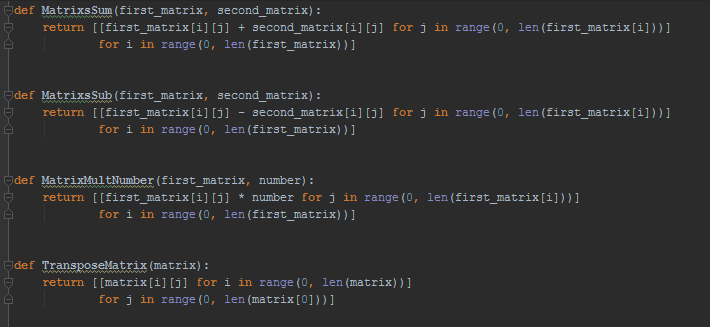
4.3 Разработка математического модуля

Создание отдельного модуля для работы с матрицами позволит неоднократно использовать данный код, а также развивать данный модуль, как отдельный проект.

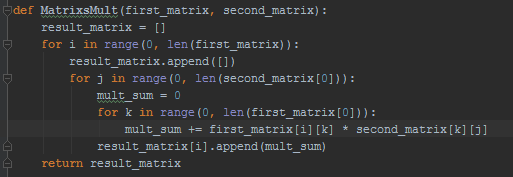
В разрабатываемом математическом модуле следует реализовать следующие методы для работы с матрицами:

* Сложение матриц
* Вычитание матриц
* Умножение матрицы на число
* Транспонирование
* Умножение матриц
* Нахождение определителя матрицы

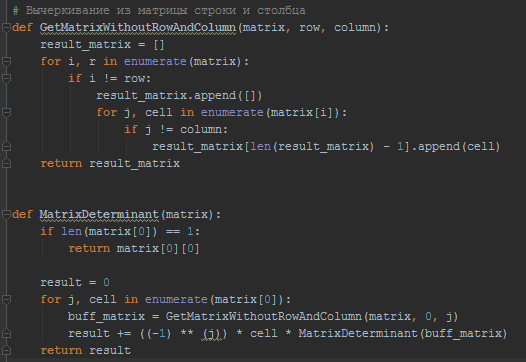
Первые четыре метода решаются с помощью такого инструмента в Python, как «генератор». Данный инструмент позволяет создавать простые массивы в Python одной строкой кода. Первые четыре реализованных метода списка можно посмотреть на рисунке \*\*\*. Последний отличается тем, что проходит сначала по строкам, а затем по столбцам, в отличии от других методов.



Метод умножения матриц требует уже более сложных вычислений, поэтому, ради читабельности кода следует использовать ручное создание матрицы в цикле с помощью метода append. Для того, чтобы создать результирующую матрицу потребуется минимум два вложенных цикла, так же. Потребуется еще один цикл, чтобы посчитать сумму умножений элементов строки первой матрицы, на элементы столбца второй. Таким образом, метод умножения матриц будет выглядеть, как на картинке \*\*\*.



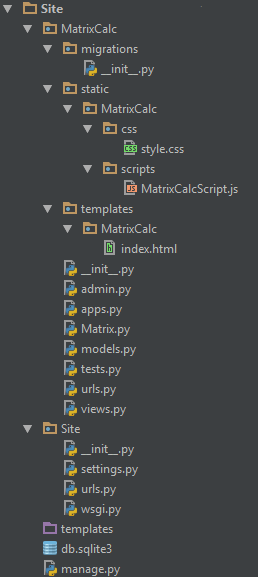
Последним методом для реализации будет метод нахождения определителя квадратной матрицы рекурсивным методом. Данный метод должен состоять из двух частей. Первой частью метода обязательно должна быть часть рекурсивного выхода, а вторая – рекурсивного вызова. Однако, перед тем, как ее создать следует выделить отдельно метод, который составляет новую матрицу из исходной, но без указанных строки и столбца. Это позволит записать рекурсивную функцию более читабельно, как на рисунке \*\*\*.



**5. Описание программного продукта**

* 1. Структура проекта

На рисунке \*\*\* изображена файловая структура проекта.



В целом, данная структура не была изменена с момента создания проекта, за исключением файла с математическими операциями. Данная структура была достигнута благодаря тому, что PyCharm IDE поочередно выполнила две команды: startproject и startapp. На первом шаге была создана следующая структура:

* Корневая директория Site/ – это просто контейнер для вашего проекта. Его название никак не используется Django, можно переименовать его во что угодно
* manage.py: Скрипт, который позволяет взаимодействовать с проектом Django
* Внутренний каталог Site/ - это пакет Python проекта. Его название – это название пакета Python, которое используется для импорта чего-либо из проекта (например, Site.urls)
* Site/\_\_init\_\_.py: Пустой файл, который указывает Python, что текущий каталог является пакетом Python
* Site/settings.py: Настройки/конфигурация проекта
* Site/urls.py: Конфигурация URL-ов для проекта Django. Это “содержание” всех Django-сайтов
* Site /wsgi.py: Точка входа проекта для WSGI-совместимых веб-серверов

На следующем шаге, командой startapp, был создан каталог внутри корневой папки с файлами проекта со следующим содержимым:

* \_\_init\_\_.py: Пустой файл, который указывает Python, что текущий каталог является пакетом Python
* migrations/: Директория, в которой сохраняются миграции
* static/: Папка со статическими файлами приложения (скриптами, файлами стилей и т.д.)
* templates/: Папка с шаблонами
* apps.py: Модуль, который добавляет в проект текущее приложение
* forms.py, models.py, admin.py, test.py, urls.py, views.py: Файлы, которые должны содержать основную логику приложения. В проекте были редактированы только последние два, а остальные не несут никакой логической нагрузки.
  1. Описание объектов и их взаимодействия

С учетом структуры расширения и используемых API можно составить общую схему взаимодействия объектов в расширении, как на рисунке \*\*\*:



Как можно заметить на рисунке \*\*\*, расширение выполняется внутри процесса браузера и содержит два основных скрипта, которые взаимодействуют между собой через объект extension. По мимо этого, фоновый скрипт расширения взаимодействует с VK API и локальным хранилищем браузера.

* 1. Описание запросов

Запросы в расширении были использованы для работы с VK API. Для создания запросов была использована функция createVkApiRequest (рисунок \*\*\*), а для работы с ними был использован обьект XMLHttpRequest.

Объект XMLHttpRequest (или, сокращенно, XHR) дает возможность браузеру делать HTTP-запросы к серверу без перезагрузки страницы. Несмотря на слово XML в названии, XMLHttpRequest может работать с данными в любом текстовом формате, и даже c бинарными данными. Его можно использовать как синхронным, так и асинхронным способом. За это отвечает третий параметр в функции send. Асинхронный вариант работы функции позволяет уменьшить «заморозки» в работе расширения. Пример работы данного обьекта можно наблюдать на рисунке \*\*\*.

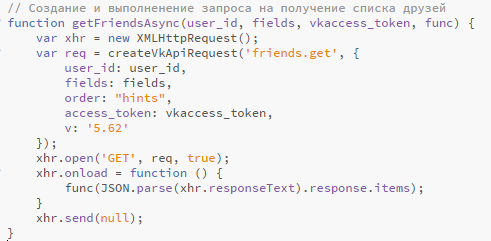
Рассмотрим детальнее запросы к VK API. Общими параметрами при любом запросе являются:

* «v» - версия API, к которой происходит запрос.
* «access\_token» - уникальный ключ пользователя, который позволяет определить личность отправителя.

Запрос на получение друзей пользователя вызывается вместе с параметрами:

* «user\_id» - id пользователя.
* «fields» - массив из названий полей, которые нужно извлечь (в нашем случае это только поле photo\_50).
* «vkacces\_token» - уникальный ключ пользователя, который позволяет определить личность отправителя.
* «order» - порядок, в котором будет отсортирован список из друзей («hints» - порядок по популярности).

Реализация данного запроса изображена на рисунке \*\*\*.



Далее рассмотрим загрузку изображения в сообщение. Первый этап – это загрузка изображения в формате blob и получение адреса сервера Вконтакте для дальнейшей загрузки с помощью метода «getMessagesUploadServer». Второй этап – это загрузка данных на сервер Вконтакте и при удачной загрузке – сохранение результатов с помощью метода «saveMessagesPhoto». После успешного сохранения изображения его можно прикреплять к сообщениям, как строку вида photo<idВладельца>\_<IdТранзакцииСохранения> к полю «attachment».

**6. Тестирование и внедрение**

6.1 Тестирование разработанного ПО

Назначение, область применения и требования к надежности программного продукта показаны в таблице \*\*\*, а результаты анализа надежности в таблице \*\*\*.

Таблица \*\*\* – Назначение, область применения и требования к надежности

|  |  |  |
| --- | --- | --- |
| Назначение | Область применения | Требования к надежности |
| Произведение основных математических операций над матрицами | Вычисления | Точный результат, стабильная работа |

Таблица \*\*\* - Анализ надежности программного продукта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № Теста | Данные для теста  (вводимая информация) | Время функционирования программы | Результат (успешно\отказ) | Результат |
|  | Открытие сайта | Меньше секунды | успешно | Открытие сайта в браузере |
|  | Выбор математической операции | Моментально | успешно | Изменение содержимого веб-страницы и изменение размеров матриц (при необходимости) |

Продолжение таблицы \*\*\*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Изменение значений в полях, отвечающих за размеры матриц | Менее секунды | успешно | Изменение размеров матриц при необходимости, ячейки матриц инициализируются нулями |
|  | Нажатие на кнопку «Посчитать» | Менее секунды | успешно | Заполнение результирующей матрицы или отображение числового значения, если выставлено нахождение определителя |

\*\*\* бла-бла Chrome инструменты для отладки

* 1. Установка программы

Основной платформой для развертывания Django является WSGI, это фактически стандарт для веб-серверов и приложений на Python.

Команда startproject установит простую WSGI конфигурацию по-умолчанию, которую можно впоследствии изменить под нужды проекта и использовать с любым WSGI-совместимым веб-сервером.

Одна из ключевых концепций развертывания с WSGI заключается в указании функции, или вызываемого объекта, application, который использует веб-сервер для взаимодействия с кодом. Обычно это объект application модуля Python доступного для сервера.

Команда startproject создаст файл <project\_name>/wsgi.py, который содержит вызываемый объект application. Этот файл использует как встроенным сервером для разработки, так и на боевом WSGI сервере.

WSGI сервер получает путь к объекту application из своих настроек. Встроенный сервер Django, запускаемый командами runserver и runfcgi, использует настройку WSGI\_APPLICATION. По умолчанию она равна <project\_name>.wsgi.application, и указывает на объект application в <project\_name>/wsgi.py.

Когда WSGI сервер загружает ваше приложение, Django необходимо импортировать модуль с настройками. Django использует переменную окружения DJANGO\_SETTINGS\_MODULE для определения расположения модуля настроек. Она должна содержать путь для импорта этого модуля. Если переменная не определена, wsgi.py использует значение mysite.settings, где mysite название вашего проекта.

В данной курсовой работе будет рассмотрен самый популярный вариант развёртывания - Apache и mod\_wsgi. Mod\_wsgi является модулем веб-сервера Apache, который может взаимодействовать с любым приложением Python, в том числе Django. Django работает с любой версией Apache, поддерживающей mod\_wsgi.

После установки и активации mod\_wsgi нужно отредактировать файл httpd.conf веб-сервера Apache, изменив его следующим образом (рисунок \*\*\*). Если используется Apache версии ниже чем 2.4, то следует заменить Require all granted` на Allow from all и добавить выше Order deny,allow.



Значение WSGIScriptAlias указывает местоположение приложений, (/ обозначает корневую директорию), вторым значением указывается расположение файла “WSGI (как правило, в корне проекта). Эти настройки позволят Apache обрабатывать любой запрос из директории, указанной как базовая с помощью WSGI-приложения, хранящегося в ней. WSGIPythonPath гарантирует, что проект доступен для импорта; иначе говоря, что команда import Site сработает. Значение <Directory> просто предоставляет Apache доступ к файлу wsgi.py.

**ЗАКЛЮЧЕНИЕ**

В результате работы над курсовой работой выполнены все поставленные задачи. Разработано работоспособное расширение для браузера Google Chrome, которое позволяет передавать информацию в социальную сеть «Вконтакте». Проанализированы существующие аналоги, выделены базовые критерии оценки. Был изучен ряд технологий для создания проекта: C HTML, CSS, Java Script, Chrome JS APIs, VK API, Brackets, json. Также, было написано обоснование выбора данных технологий в курсовом проекте. Функционал расширения был утвержден согласно разработанной структуре программы. На основе разработанной функциональной схемы проекта составлен программный продукт, обладающий интерфейсной частью и работой со сторонними API.

На данный момент при работе с расширением доступны следующие возможности: добавление аккаунта социальной сети в список аккаунтов расширения, изменение текущего активного аккаунта и удаление аккаунтов из списка расширения, изменение списка адресатов для текущего аккаунта, отправка разных типов данных в социальные сети.

Дальнейшее развитие программы связано с расширением ее возможностей, улучшением уровня безопасности, добавление ее в магазин Google Extensions. Тестирование программы показало ее работоспособность и высокую степень надежности.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Интернет-магазин Chrome

URL: https://chrome.google.com/webstore/ (дата обращения: 14.02.2017)

1. Официальная страница фреймворка Bootstrap 3

URL: http://getbootstrap.com/ (дата обращения: 15.02.2017)  
 3. Wikipedia – информационный ресурс:

URL: https://ru.wikipedia.org/wiki/HTML5

(дата обращения: 15.02.2017)

URL: https://ru.wikipedia.org/wiki/JavaScript

(дата обращения: 15.02.2017)

URL: https://en.wikipedia.org/wiki/Brackets\_(text\_editor)

(дата обращения: 15.02.2017)

4. Документация API Вконтакте

URL: https://vk.com/dev/manuals (дата обращения: 20.02.2017)

5. JavaScript. Подробное руководство / Дэвид Флэнаган / Год: 2013

6. w3school: информационный ресурс

URL: https://www.w3schools.com/html/default.asp  
(дата обращения: 21.02.2017)

URL: https://www.w3schools.com/css/default.asp  
(дата обращения: 22.02.2017)

URL: https://www.w3schools.com/js/default.asp  
(дата обращения: 25. 02.2017)  
 7. Chrome Developer // [ресурс с материалами для разработки] //

URL: https://developer.chrome.com/extensions (дата посещения: 03.03.2017)

## ПРИЛОЖЕНИЕ А

(обязательное)

**Фрагменты листинга**

Листинг А.1 Файл background.js

var debug = true,

vkCLientId = '5886692',

vkRequestedScopes = 'docs,offline,messages,wall,photos',

vk\_default\_redirect\_uri = 'https://oauth.vk.com/blank.html',

files\_to\_share = [{

title: "Изображение",

context: ["image"]

},

{

title: "Видео",

context: ["video"]

},

{

title: "Ссылка",

context: ["link"]

},

{

title: "Страница",

context: ["all"]

},

{

title: "Текст",

context: ["selection"]

}];

function log(msg) {

if (debug)

console.log(msg);

}

function logObj(msg, obj) {

if (debug)

console.log(msg, obj);

}

chrome.extension.onConnect.addListener(function (port) {

log("Connected .....");

port.onMessage.addListener(function (msg) {

log("message recieved " + msg.message);

switch (msg.message) {

case "getFriends":

{

chrome.storage.local.get([

'users\_dic',

'vk\_user\_id'

], function (items) {

if (items.vk\_user\_id != undefined) {

log(items.vk\_user\_id);

log(items.users\_dic);

getFriendsAsync(items.vk\_user\_id, "photo\_50", items.users\_dic[items.vk\_user\_id].vkaccess\_token, function (friends) {

var mod\_friends = friends.map(function (friend) {

items.users\_dic[items.vk\_user\_id].sub\_friends.forEach(function (sub\_friend) {

if (friend.id == sub\_friend.user\_id) {

friend.checked = "checked";

}

//log(friend.user\_id + sub\_friend.user\_id)

});

return friend;

});

msg = {

message: "returnFriends",

friends: mod\_friends,

}

port.postMessage(msg);

});

} else {

msg = {

message: "returnFriends",

friends: {},

}

port.postMessage(msg);

}

});

}

break;

case "getAccounts":

{

chrome.storage.local.get([

'users\_dic',

'vk\_user\_id'

], function (items) {

var accounts = [];

for (var key in items.users\_dic) {

if (items.vk\_user\_id == key) {

items.users\_dic[key].checked = "checked";

}

items.users\_dic[key].id = key;

accounts.push(items.users\_dic[key]);

}

var response = {

message: "returnAccounts",

accounts: accounts,

}

port.postMessage(response);

});

}

break;

case "ChangeFriend":

{

log(msg.user\_id + " " + msg.checked);

chrome.storage.local.get([

'users\_dic',

'vk\_user\_id'

], function (items) {

if (msg.checked == true) {

items.users\_dic[items.vk\_user\_id].sub\_friends.push({

'user\_id': msg.user\_id,

'user\_full\_name': msg.user\_full\_name

});

} else {

items.users\_dic[items.vk\_user\_id].sub\_friends = items.users\_dic[items.vk\_user\_id].sub\_friends.filter(function (e) {

return JSON.stringify(e) !== JSON.stringify({

'user\_full\_name': msg.user\_full\_name,

'user\_id': msg.user\_id

});

});

}

chrome.storage.local.set({

'users\_dic': items.users\_dic,

}, function () {

log("Friend sub list changed to - " + items.users\_dic[items.vk\_user\_id].sub\_friends.toString());

UpdateContextMenu();

});

});

}

break;

case "SetCurrentUser":

{

log(msg.user\_id + " " + msg.checked);

chrome.storage.local.set({

'vk\_user\_id': msg.user\_id,

}, function () {

log("Current user changed to - " + msg.user\_id);

UpdateContextMenu();

var response = {

message: "userChange",

}

port.postMessage(response);

});

}

break;

case "refreshUser":

{

vkAuthorizationDialog(function () {

UpdateContextMenu();

var response = {

message: "userRefresh",

}

port.postMessage(response);

});

}

break;

case "deleteUser":

{

log(msg.user\_id);

chrome.storage.local.get(['users\_dic', 'vk\_user\_id'],

function (result) {

delete result.users\_dic[msg.user\_id];

log(result.vk\_user\_id);

if (result.vk\_user\_id == msg.user\_id) {

chrome.storage.local.remove('vk\_user\_id');

}

log(result);

chrome.storage.local.set({

'users\_dic': result.users\_dic

}, function () {

UpdateContextMenu();

var response = {

message: "userDelete",

}

port.postMessage(response);

});

});

}

break;

}

});

});

// Вытягиваем параметры из адресной строки

function getUrlParameterValue(url, parameterName) {

"use strict";

var urlParameters = url.substr(url.indexOf("#") + 1),

parameterValue = "";

urlParameters = urlParameters.split("&");

for (var index = 0; index < urlParameters.length; index += 1) {

var temp = urlParameters[index].split("=");

if (temp[0] === parameterName) {

return temp[1];

}

}

return parameterValue;

}

// Создание запроса

function createRequest(path, params) {

let second\_part = [];

for (let d in params) {

second\_part.push(encodeURIComponent(d) + '=' + encodeURIComponent(params[d]));

}

return path + '?' + second\_part.join('&');

}

// создание запроса к api.vk.com

function createVkApiRequest(method\_name, params) {

var first\_part = 'https://api.vk.com/method/' + method\_name;

return createRequest(first\_part, params);

}

Листинг А.2 Файл popup.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

<!-- Latest compiled and minified CSS -->

<link rel="stylesheet" href="bootstrap-3.3.7-dist/bootstrap-3.3.7-dist/css/bootstrap.css">

<!-- jQuery library -->

<script src="scripts/jquery-3.1.1.min.js"></script>

<!-- Latest compiled JavaScript -->

<script src="bootstrap-3.3.7-dist/bootstrap-3.3.7-dist/js/bootstrap.js"></script>

<link rel="stylesheet" href="style.css">

<script src="scripts/mustache.js"></script>

<script src="scripts/popup.js"></script>

</head>

<body>

<ul class="nav nav-tabs">

<li class="active"><a data-toggle="tab" href="#accounts\_tab">Аккаунты</a></li>

<li><a data-toggle="tab" href="#friends\_tab">Адресаты</a></li>

<!-- <li><a data-toggle="tab" href="#settings\_tab">Настройки</a></li> -->

</ul>

<div class="tab-content">

<div id="accounts\_tab" class="tab-pane fade in active">

<button id="refresh\_user" type="button" class="btn btn-primary btn-block"> Добавить текущий аккаунт </button>

<hr>

<script id="account\_template" type="text/x-custom-template">

<label>

<input class="check\_account" type="radio" name="account\_radio" value="{{id}}" style="width:50px;height:50px" {{checked}} hidden>

<div class="account\_row">

<div class="account row">

<div class="col-xs-9">

<p>{{first\_name}}</p>

<p>{{last\_name}}</p>

</div>

<div class="col-xs-3">

<button class="delete\_account btn btn-danger" value="{{id}}" style="width:50px;height:50px;margin-top:4px;">X</button>

</div>

</div>

<div>

</label>

<hr>

</script>

<div id="accounts">

</div>

</div>

<div id="friends\_tab" class="tab-pane fade">

<script id="friend\_template" type="text/x-custom-template">

<label>

<div class="friend row">

<div class="col-xs-3">

<img src="{{photo\_50}}" class="img-circle">

</div>

<div class="col-xs-6">

<p>{{first\_name}}</p>

<p>{{last\_name}}</p>

</div>

<div class="col-xs-3">

<input class="check\_friend" type="checkbox" value="{{last\_name}} {{first\_name}} {{id}}" style="width:50px;height:50px" {{checked}}>

</div>

</div>

</label>

<hr>

</script>

<div id="friends">

</div>

</div>

<div id="settings\_tab" class="tab-pane fade">

</div>

</div>

</body>

</html>

Листинг А.3 Файл popup.js

window.onload = function () {

var port = chrome.extension.connect({

name: "Sample Communication"

});

port.onMessage.addListener(function (msg) {

switch (msg.message) {

case "returnFriends":

{

var friends\_list = document.getElementById("friends");

var friend\_template = document.getElementById("friend\_template");

friends\_list.innerHTML = "";

var friendAdd = function (target, template, friend) {

var friend\_li = Mustache.render(template.innerHTML, friend);

target.innerHTML += (friend\_li);

};

msg.friends.forEach(function (friend, i) {

friendAdd(friends\_list, friend\_template, friend);

});

$(".check\_friend").change(function (target) {

var full\_name = this.value.split(' ').slice(0, 2).join(' '),

user\_id = this.value.split(' ')[2];

var msg = {

message: "ChangeFriend",

user\_full\_name: full\_name,

user\_id: user\_id,

checked: this.checked,

}

port.postMessage(msg);

});

}

break;

case "returnAccounts":

{

var accounts\_list = document.getElementById("accounts");

var account\_template = document.getElementById("account\_template");

accounts\_list.innerHTML = "";

msg.accounts.forEach(function (account) {

var account\_li = Mustache.render(account\_template.innerHTML, account);

accounts\_list.innerHTML += (account\_li);

});

getFriends();

$(".check\_account").change(function (target) {

user\_id = this.value;

var msg = {

message: "SetCurrentUser",

user\_id: user\_id,

}

port.postMessage(msg);

});

}

break;

case "userDelete":

case "userRefresh":

case "userChange":

{

getAccounts();

}

break;

}

});

var getAccounts = function () {

port.postMessage({

message: "getAccounts"

});

}

var getFriends = function () {

port.postMessage({

message: "getFriends"

});

}

getAccounts();

$("body").on("click", ".delete\_account", function (target) {

user\_id = this.value;

port.postMessage({

message: "deleteUser",

user\_id: user\_id

});

});

$("#refresh\_user").click(function () {

port.postMessage({

message: "refreshUser"

});

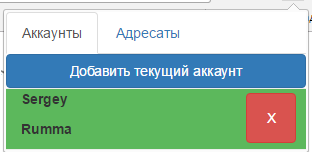
});

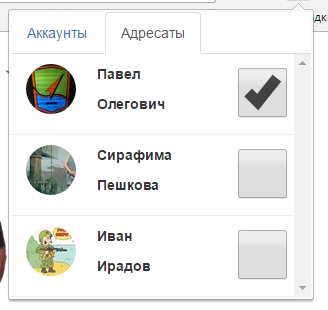
}

## ПРИЛОЖЕНИЕ Б

(обязательное)

**Экранные формы**



****