**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**Департамент анализа данных,**

**принятия решений и финансовых технологий**

**Пояснительная записка к междисциплинарной курсовой работе**

на тему:

**Разработка интерфейса для сервиса нормализации списка ФИО**

Выполнил:

студент группы ПИ2-1

Носко Максим Михайлович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

Научный руководитель:

Догучаева Светлана Магомедовна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

2019

Содержание

[Введение 3](#_Toc7852954)

[1. Постановка задачи 4](#_Toc7852955)

[2. Описание предметной области 5](#_Toc7852956)

[3. Актуальность автоматизации 5](#_Toc7852957)

[4. Описание программы 6](#_Toc7852958)

[4.1. Получение исходных данных от клиентской части приложения 6](#_Toc7852959)

[4.2. Нормализация данных 7](#_Toc7852960)

[4.3. Анализ результатов нормализации 8](#_Toc7852961)

[4.4. Отправка нормализованных данных клиентской части приложения 8](#_Toc7852962)

[4.5. Предоставление пользователю веб-интерфейса для ввода исходных данных 8](#_Toc7852963)

[4.6. Предоставление пользователю веб-интерфейса для ввода шаблона 9](#_Toc7852964)

[4.7. Демонстрация пользователю справочной информации относительно возможностей и особенностей использования шаблонов 9](#_Toc7852965)

[4.8. Отправка исходных данных на сервер 10](#_Toc7852966)

[4.9. Получение нормализованных данных от сервера 11](#_Toc7852967)

[4.10. Приведение нормализованных данных к заданному пользователем шаблону 11](#_Toc7852968)

[4.11. Предоставление пользователю интерфейса для работы с результатами нормализации и приведения к шаблону 12](#_Toc7852969)

[4.12. Состав приложения 14](#_Toc7852970)

[5. Назначение и состав файлов программы 15](#_Toc7852971)

[5.1. Диаграмма объектов 15](#_Toc7852972)

[5.2. Файл app.js – основной файл клиентской части приложения 16](#_Toc7852973)

[5.3. Файл server.py – основной файл серверной части 18](#_Toc7852974)

[Заключение 18](#_Toc7852975)

[Список литературы 19](#_Toc7852976)

[Приложение. Исходный код программы 19](#_Toc7852977)

# Введение

Целью курсовой работы является приобретение навыков в разработке прикладного web-приложения, состоящего из двух частей, работающих вместе для решения поставленной задачи:

* Серверной части, выполняющей основную обработку данных, с использованием следующих технологий:
* Язык программирования Python – основной язык для разработки, в данном случае, серверной части приложения с использованием подключаемых фреймворков и библиотек
* Фреймворка Flask, позволяющего создавать веб-сервера на Python
* Библиотека нормализации ФИО «Algorithm\_FIO»[[1]](#footnote-1)
* Клиентской части, представляющей из себя веб-интерфейс, с использованием следующих технологий:
* Язык программирования JavaScript – основной язык для разработки интерактивного содержимого с использованием подключаемых фреймворков и библиотек
* Фреймворк jQuery
* Фреймворк React – использование данного фреймворка было необходимо для быстрой разработки интерактивного веб-интерфейса
* Библиотека Material Design – использование данной библиотеки было необходимо для внедрения в интерактивный React-веб-интерфейс готовых решений в области внешнего вида элементов интерфейса
* Язык CSS (от англ. Cascading Style Sheets – «каскадные таблицы стилей») – язык таблиц стилей, обеспечивающий удобное для разработки управление стилями – то есть внешним видом элементов интерфейса
* Язык разметки HTML (от англ. HyperText Markup Language – «язык гипертекстовой разметки») – язык разметки, лежащий в основе элементов веб-интерфейса.
* Менеджер пакетов NPM

Второй целью курсовой работы, и не менее важной, является разработка пояснительной записки, отвечающей таким требованиям, как полнота, лаконичность, техническая корректность и т.д. Техническая документация является составной частью программного продукта и, следовательно, определяет его качество. Поэтому приобретение навыков в разработке технической документации, а в данном случае пояснительной записки, также является важной задачей.

# Постановка задачи

В соответствии с выбранной темой требуется разработать серверную часть web-приложения, получающую набор исходных данных от клиентской части через HTTP-протокол, производящую их нормализацию (то есть приведение произвольно записанного ФИО к стандарту «Фамилия Имя Отчество») и возвращающую клиентской части через этот же протокол набор уже нормализованных ФИО. Клиентская часть, соответственно, должна обеспечивать отправку исходных данных на сервер, их получение и приведение нормализованных данных к задаваемому пользователем шаблону.

Требования:

* Разрабатываемое web-приложение должно представлять собой серверную и клиентскую части
* Серверная часть взаимодействует с клиентской посредством HTTP
* Пользователь взаимодействует только с клиентской частью, то есть с web-интерфейсом
* Ввод исходных данных, ввод шаблона и отображение результата происходит на клиентской стороне
* Нормализация ФИО происходит на серверной стороне приложения
* Приведение нормализованных ФИО к заданному шаблону осуществляется клиентской стороной приложения
* Программа не должна завершаться аварийно

# Описание предметной области

В разрабатываемом приложении предметной областью автоматизации является нормализация ФИО вместе с приведением набора ФИО к необходимому пользователю шаблону.

# Актуальность автоматизации

На текущий момент для приведения ненормализованного набора ФИО к нужному шаблону пользователю необходимо выполнять обе эти операции вручную:

* Для нормализации ФИО необходимо сначала исправить ошибки в исходных данных, в чём могут помочь вспомогательные системы исправления ошибок в тексте, которые есть в популярных браузерах и в текстовых процессорах, реже – в текстовых редакторах. Но для завершения нормализации необходимо также нормализовать порядок следования слов в ФИО, в чём уже вышеупомянутые программы не могут помочь, вследствие чего это приходится делать пользователю вручную, что занимает немало времени на каждую запись
* После нормализации пользователю необходимо вручную привести каждую запись к нужному стандарту, что также будет занимать немало времени на каждую запись.

После автоматизации планируется, что пользователь введёт в приложение исходные данные и шаблон, а приложение выдаст пользователю уже нормализованные и приведённые к необходимому шаблону ФИО. Таким образом пользователь сможет решить данную задачу значительно быстрее, что делает задачу автоматизации актуальной.

# Описание программы

В разрабатываемом приложении серверная часть имеет следующие задачи:

* Получение исходных данных от клиентской части приложения
* Нормализация данных
* Анализ результатов нормализации
* Отправка нормализованных данных клиентской части приложения

Клиентская часть разрабатываемого приложения имеет следующие задачи:

* Предоставление пользователю веб-интерфейса для ввода исходных данных
* Предоставление пользователю веб-интерфейса для ввода шаблона
* Демонстрация пользователю справочной информации относительно возможностей и особенностей использования шаблонов
* Отправка исходных данных на сервер
* Получение нормализованных данных от сервера
* Приведение нормализованных данных к заданному пользователем шаблону
* Предоставление пользователю интерфейса для работы с результатами нормализации и приведения к шаблону
* Сообщение пользователю об успешности нормализации

# Получение исходных данных от клиентской части приложения

Для получения исходных данных от клиентской части используется POST-запрос в рамках протокола HTTP.

Для непосредственного взаимодействия с HTTP, то есть, для получения запросов и отправки ответов на них, используется фреймворк Flask, который позволяет, используя язык программирования Python, развернуть веб-сервер, который и позволяет работать с POST-запросами.

Тип запроса POST был выбран из-за того, что он позволяет передавать сериализованные текстовые данные достаточно большой длины, что и выполняется при работе приложения: на сервер приходит POST-запрос, содержащий в себе переменную, доступную по ключу “str[]”, содержащую в себе список исходных ФИО.

Пример POST-данных, получаемых от клиента:

{'str[]': ['Иванов Иван', 'Максим Михайлович']}

# Нормализация данных

Извлечённые из запроса данные в виде списка отправляются на нормализацию в качестве аргумента функции NotBruteAtAll из библиотеки Algorithm\_FIO.

Функция возвращает нормализованные данные в определённом формате, для каждой записи возвращая следующие данные:

* Нормализованную запись ФИО
* Отметки о внесении исправлений в запись
* Порядок следования и состав записи
  + 0 – фамилия
  + 1 – имя
  + 2 – отчество
* Пол, вычисленный по содержимому записи
  + М – мужской
  + Ж – женский
  + МЖ – в случае, если анализ определения пола не дал определённого результата, что означает, что запись не удалось нормализовать

Пример данных, которые возвращает функция NotBruteAtAll:

[[['Иванов Иван', 'М'], [[0, '']]], [0, 1], [['Носко Максим', 'М'], [[0, '']]], [1, 0]]

Где [[0, '']] означает отсутствие исправлений, а [1,0] – порядок следования имени частей ФИО в исходных данных

# Анализ результатов нормализации

Для приведения данных к заданному пользователем шаблону необходимо, после нормализации, произвести анализ её результатов, создав для каждой записи словарь со следующими ключами:

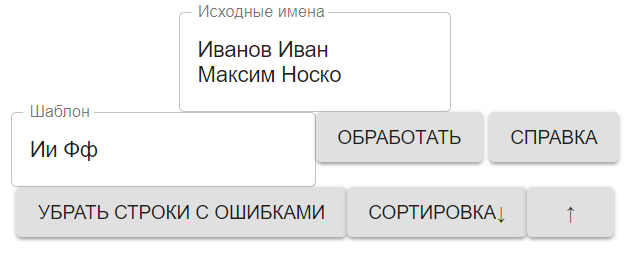
* Имя: если алгоритм вернул соответствующие данные, то строка с нормализованным именем, если нет, то None
* Фамилия: если алгоритм вернул соответствующие данные, то строка с нормализованным именем, если нет, то None
* Отчество: если алгоритм вернул соответствующие данные, то строка с нормализованным именем, если нет, то None
* Пол: значение, соответствующее результатам нормализации
* Успех: отметка о том, были ли внесены исправления

# Отправка нормализованных данных клиентской части приложения

После анализа результатов нормализации производится создание JSON-строки, содержащий список результатов нормализации для каждой записи, с сохранением порядка следования записей в исходных данных

# Предоставление пользователю веб-интерфейса для ввода исходных данных

На веб-странице, которая генерируется с использованием фреймворка React, располагается многострочное текстовое поле (см. рис.1), куда пользователь может ввести исходные данные или вставить их из буфера обмена. Данное поле выполнено с использованием библиотеки Material Design, оно также содержит в себе краткое описание своего назначения и подсказку о формате вводимых данных



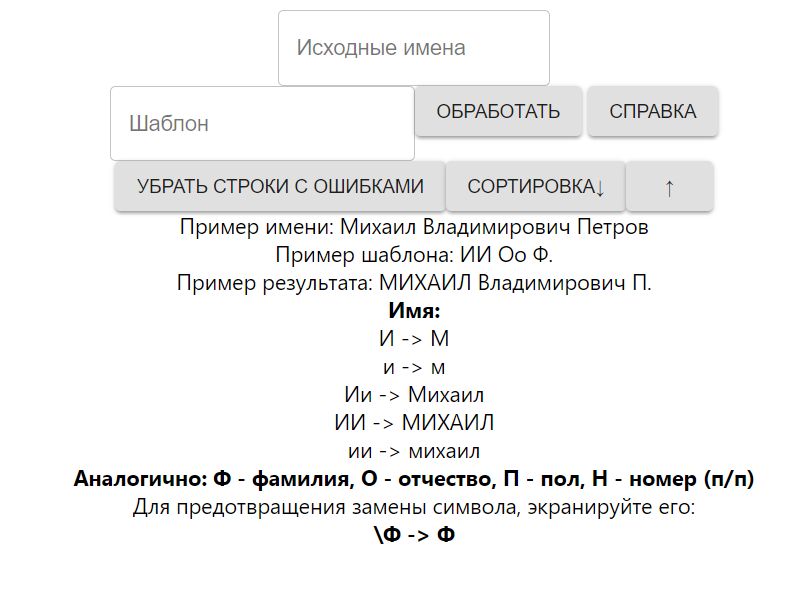
**Рис. 1:** веб-интерфейс приложения после ввода данных

# Предоставление пользователю веб-интерфейса для ввода шаблона

На той же веб-странице, ниже располагается однострочное текстовое поле для ввода шаблона(см. Рис.1), к которому будут приводиться нормализованные ФИО, полученные от сервера. Данное текстовое поле также выполнено с использованием Material Design и содержит в себе краткое описание своего назначение и подсказку о формате вводимых данных

# **Демонстрация пользователю справочной информации** относительно возможностей и особенностей использования шаблонов

Когда пользователь нажимает кнопку «Справка», то программа выводит на страницу справочную информацию относительно использования шаблонов, как это продемонстрировано на рисунке 2.



**Рис.2:** вывод справочной информации

# Отправка исходных данных на сервер

Когда пользователь нажимает кнопку «Обработать», то программа извлекает значение текстового поля, предназначенного для ввода исходных данных и, с помощью функции ajax фреймворка jQuery совершает POST-запрос по протоколу HTTP на сервер, передавая серверу исходные данные:

$.ajax

(

{

type:"POST",

url: "http://localhost:8765/",

data: {"str":document.getElementById("txt\_vvod").value.split("\n")},

success:otvet

}

)

# Получение нормализованных данных от сервера

В случае, если POST-запрос успешно был совершён, от сервера будут получены данные в формате JSON (см. пункт 4.4), которые десериализуются в список, после чего передаются функции, отвечающей за приведение нормализованных данных к шаблону.

# Приведение нормализованных данных к заданному пользователем шаблону

Приведение нормализованных данных к шаблону производится построчно и независимо для каждой возвращённой сервером записи.

Для этого шаблон анализируется посимвольно, и разбивается на группы, следующие в шаблоне друг за другом. Каждая группа является списком и содержит в себе информацию о том, будет ли она заменена на соответствующую часть нормализованной записи, полученной от сервера, или же будет оставлена в исходном виде. Данная операция производится с учётом экранирования символов в шаблоне: такие символы не будут заменены на соответствующие им части нормализованного ФИО.

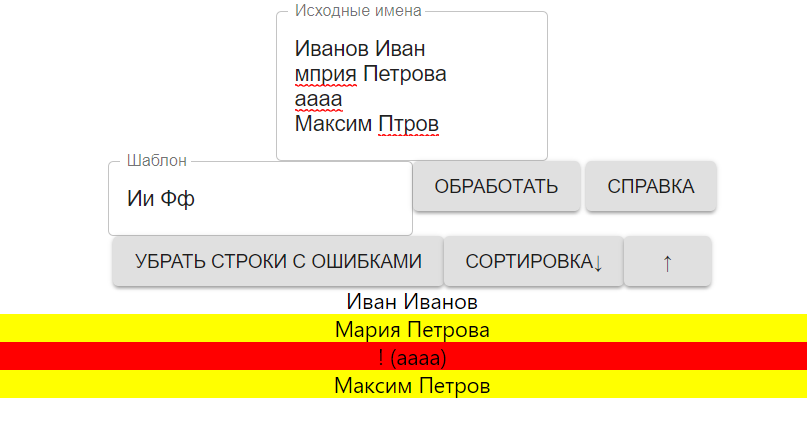
# Предоставление пользователю интерфейса для работы с результатами нормализации и приведения к шаблону

После того, как нормализованные данные были приведены к заданному пользователем шаблону, они выводятся на веб-страницу, с отметками об успешности нормализации с помощью цвета фона (см. рис.3):

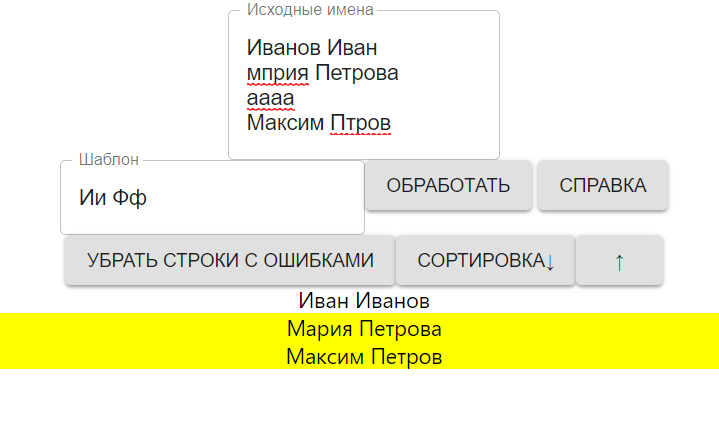
* На белом фоне выводятся ФИО, которые
* Уже были нормализованы в исходных данных
* Отличались от нормы только регистром и/или порядком слов
* На жёлтом фоне выводятся ФИО, которые
* Содержали в себе небольшие ошибки, которые удалось исправить
* На красном фоне выводятся ФИО, которые
* Содержали в себе ошибки, которые не удалось исправить, вследствие чего работать дальше с такими данными невозможно

Для ускорения работы с данными пользователь может нажать кнопку «Убрать строки с ошибками», которая удалит все ФИО на красном фоне (см. рис.4).

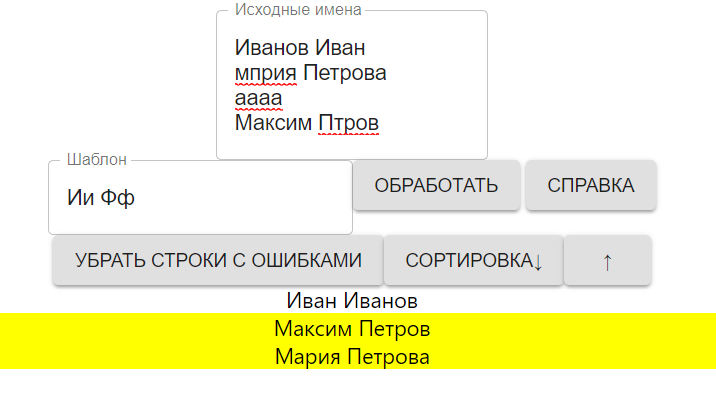
Также, пользователю доступна сортировка ФИО-результатов как в лексикографическом порядке (см. рис.5), так и в порядке, обратном лексикографическому, что позволяет, применяя различные шаблоны, получить итоговые данные не только в порядке их ввода в качестве исходных данных, но и, например, сортировку по именам, фамилиям или по полу (см. рис.6).



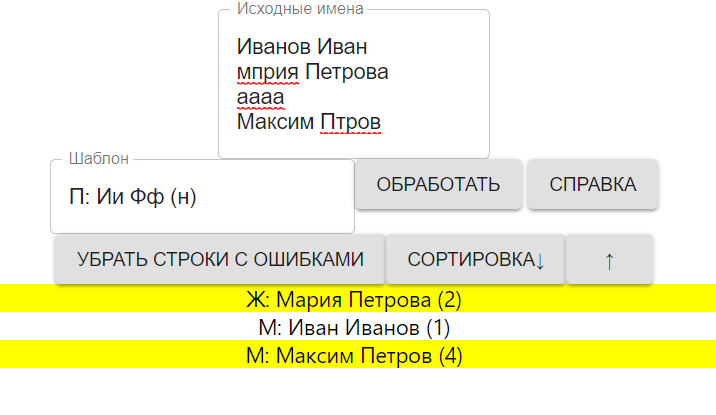
**Рис.3:** вывод результата



**Рис.4:** очистка результата от бесполезных записей



**Рис.5:** сортировка результирующих записей в лексикографическом порядке



**Рис.6:** пример использования полового признака записи для разделения по нему при сортировке записей

# Состав приложения

В состав серверной части входят следующие файлы:

* Server.py – файл с исходным кодом серверной части приложения на языке Python
* FIOAlgorithm – папка с библиотекой «Algorithm\_FIO»

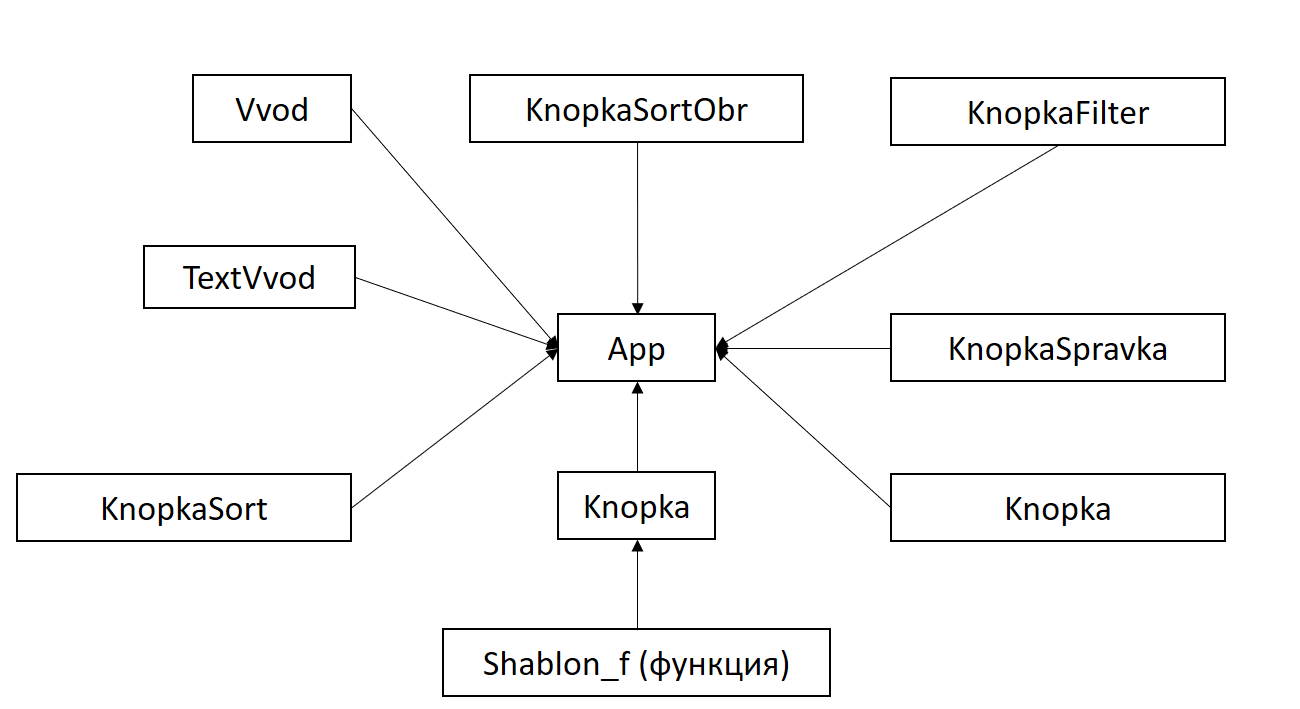
В состав клиентской части входят следующие файлы:

* node\_modules – папка, хранящая в себе необходимые модули, создаваемая NPM
* public – папка, хранящая в себе необходимые основные файлы react-приложения:
* index.html – основная страница приложения
* manifest.json – файл основной конфигурации приложения
* package.json – файл, содержащий зависимости приложения
* src – папка, содержащая исходный код приложения
* App.css и index.css – файлы стилей приложения
* Index.js и serviceWorker.js – необходимые файлы React-приложения

# Назначение и состав файлов программы

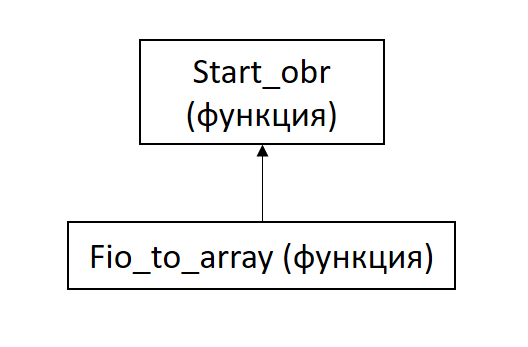
# Диаграмма объектов

Для клиентской части приложения:



**Рис.7:** диаграмма объектов клиентской части приложения

Для серверной части приложения:



**рис. 8:** диаграмма объектов серверной части приложения

# Файл app.js – основной файл клиентской части приложения

Класс App, унаследованный от Component (как и все нижеперечисленные), представляет из себя основной класс, формирующий страницу, он использует в себе все остальные классы в своём методе render:

* Класс Rezultat – для формирования контейнера для результата обработки
* Метод render возвращает контейнер
* Класс TextVvod – для формирования текстового поля ввода исходных данных
* Метод render возвращает текстовое поле
* Класс Vvod – для формирования текстового поля для ввода шаблона
* Метод render возвращает текстовое поле
* Класс Knopka – для формирования кнопки обработки данных
* Имеет в себе метод mymethod для обработки нажатия на кнопку

Данный метод проверяет наличие текста в поле ввода исходных данных и шаблона, и прекращает свою работу, если в одном или в двух из этих текстовых полей пусто.

После проверки данный метод отправляет данные на сервер, а при успешном состоянии POST-запроса вызывает внутреннюю функцию otvet.

Функция otvet получает в качестве аргумента ответ сервера в формате JSON, парсит его, после чего производит обработку каждой нормализованной записи (см. п. 4.11) и добавляет в контейнер для результатов возвращаемое значения вызова shablon\_f, куда передаёт аргументом шаблон и каждую запись из ответа сервера.

* Метод render возвращает кнопку
* Класс KnopkaSpravka – для формирования кнопки вызова справки
* Имеет в себе метод mymethod для обработки нажатия на кнопку

Данный метод заменяет текущее содержимое контейнера для результатов на справочную информацию

* Метод render возвращает кнопку
* Класс KnopkaSort – для формирования кнопки, позволяющей отсортировать результаты в лексикографическом порядке
* Имеет в себе метод mymethod для обработки нажатия на кнопку

Данный метод сортирует дочерние контейнеры внутри контейнера для результатов в лексикографическом порядке

* Метод render возвращает кнопку
* Класс KnopkaSortObr – для формирования кнопки, позволяющей отсортировать результаты в порядке, обратном лексикографическому
* Имеет в себе метод mymethod для обработки нажатия на кнопку

Данный метод сортирует дочерние контейнеры внутри контейнера для результатов в порядке, обратном лексикографическому

* Метод render возвращает кнопку
* Класс KnopkaFilter – для формирования кнопки, удаляющей бесполезные результаты
* Метод render возвращает кнопку
* Имеет в себе метод mymethod для обработки нажатия на кнопку

Данный метод удаляет со страницы все элементы класса oshybka

Также, в файле есть функция shablon\_f, которая позволяет разбить шаблон на группы и применить его для нормализованной записи:

Сначала данная функция создаёт список из значащих букв в верхнем регистре, после чего для каждого символа шаблона проверяет, с учётом экранирования, потребуется ли замена (см. п. 4.10) символа на часть нормализованного ФИО, после чего производит данную замену и возвращает результат.

# Файл server.py – основной файл серверной части

Данный файл состоит из двух функций:

* fio\_to\_array – функции, выполяющей нормализацию ФИО с использованием подключаемой библиотеки нормализаци
* start\_obr – функции, связанной с помощью декоратора с фреймворком Flask, вызывающая по входящему запросу на сервер.

# Заключение

В ходе выполнения данной работы было создано web-приложение из клиентской и серверной частей в соответствии с заявленными требованиями. Использование данного приложения позволяет нормализовать и привести к нужному шаблону набор ненормализованных ФИО.

# Список литературы

1. Документация React [Интернет-ресурс] URL: <https://reactjs.org/docs/>
2. Документация Flask [Интернет-ресурс] URL: <http://flask.pocoo.org/docs/1.0/>

# Приложение. Исходный код программы

**Серверная часть:**

**Файл server.py:**

**from** FIOAlgorithm **import** FIOalgorithm  
**def** fio\_to\_array(massiv):  
 obr\_massiv=FIOalgorithm.NotBruteAtAll(massiv)  
 rez\_m=[]  
 nomer=1  
 **for** i **in** range(0, len(obr\_massiv), 2):  
 rez\_m.append({**"Имя"**:**None**,**"Фамилия"**:**None**,**"Отчество"**:**None**,**"Пол"**:**None**,**"Правильность"**:**True**,**"Номер"**:str(nomer)})  
 nomer+=1  
 **if not**((obr\_massiv[i][1]==[[0, **''**]]) **or** (obr\_massiv[i][1]==[])):  
 rez\_m[len(rez\_m) - 1][**"Правильность"**]=**False** slova=obr\_massiv[i][0][0].split()  
 rez\_m[len(rez\_m) - 1][**"Пол"**] = obr\_massiv[i][0][1]  
 **if** rez\_m[len(rez\_m) - 1][**"Пол"**] == **"М"**:  
 rez\_m[len(rez\_m) - 1][**"Пол"**]=**"Мужской"  
 elif** rez\_m[len(rez\_m) - 1][**"Пол"**] == **"Ж"**:  
 rez\_m[len(rez\_m) - 1][**"Пол"**]=**"Женский"** obr\_massiv[i + 1].sort()  
 **for** ti,z **in** enumerate(obr\_massiv[i+1]):  
 **if** z==0:  
 rez\_m[len(rez\_m)-1][**"Фамилия"**]=slova[ti]  
 **elif** z==1:  
 rez\_m[len(rez\_m) - 1][**"Имя"**] = slova[ti]  
 **elif** z==2:  
 rez\_m[len(rez\_m) - 1][**"Отчество"**] = slova[ti]  
 **return** rez\_m  
**from** flask **import** Flask  
**from** flask **import** request  
**from** flask\_cors **import** CORS  
**from** json **import** dumps  
app = Flask(\_\_name\_\_)  
CORS(app)  
@app.route(**'/'**, methods=[**"POST"**])  
**def** start\_obr():  
 tm=dict(request.form)  
 **return** dumps(fio\_to\_array(tm[**"str[]"**]))  
app.run(host=**'0.0.0.0'**, port=8765)

**Клиентская часть:**

**Файл /package.json:**

{

"name": "FIOnormapp",

"version": "0.1.0",

"private": true,

"dependencies": {

"@material-ui/core": "^3.9.2",

"jquery": "^3.3.1",

"react": "^16.8.4",

"react-dom": "^16.8.4",

"react-scripts": "2.1.8"

},

"scripts": {

"start": "react-scripts start",

"build": "react-scripts build",

"test": "react-scripts test",

"eject": "react-scripts eject"

},

"eslintConfig": {

"extends": "react-app"

},

"browserslist": [

">0.2%",

"not dead",

"not ie <= 11",

"not op\_mini all"

]

}

**Файл /src/App.js:**

import React, { Component } from 'react';

import './App.css';

import Button from "@material-ui/core/Button";

import AppBar from '@material-ui/core/AppBar';

import ToolBar from '@material-ui/core/AppBar';

import IconButton from '@material-ui/core/AppBar';

import MenuIcon from '@material-ui/core/AppBar';

import TextField from '@material-ui/core/TextField';

import { withStyles } from '@material-ui/core/styles';

import \* as $ from 'jquery';

function shablon\_f(txt,m)

{

var mrez=[];

var symb="";

var tm=[];

for(var i in m)

{

if(m[i]!=null)

{

symb+=i[0].toUpperCase();

tm.push(m[i]);

}

else

{

symb+=i[0].toUpperCase();

m[i]=" ";

tm.push(" ");

}

}

m=tm;

var i=1;

if(txt[0]!="\\")

{

mrez.push([symb.indexOf(txt[0].toUpperCase()),txt[0]])

}

else

{

mrez.push([-1,txt[1]]);

i++;

}

for(i;i<txt.length;i++)

{

if(txt[i]!="\\")

{

if(symb.indexOf(txt[i].toUpperCase())==mrez[mrez.length-1][0])

{

mrez[mrez.length-1][1]+=txt[i];

}

else

{

mrez.push([symb.indexOf(txt[i].toUpperCase()),txt[i]])

}

}

else

{

mrez.push([-1,txt[i+1]])

i+=1;

}

}

var rez="";

for(var i=0;i<mrez.length;i++)

{

if(mrez[i][0]==-1)

{

rez+=mrez[i][1];

}

else

{

if(mrez[i][1].length==1)

{

if(mrez[i][1]==symb[mrez[i][0]])

{

rez+=m[mrez[i][0]][0];

}

else

{

rez+=m[mrez[i][0]][0].toLowerCase();

}

}

else

{

if(mrez[i][1]==(symb[mrez[i][0]]+symb[mrez[i][0]].toLowerCase()))

{

rez+=m[mrez[i][0]];

}

if(mrez[i][1]==symb[mrez[i][0]].repeat(2))

{

rez+=m[mrez[i][0]].toUpperCase();

}

if(mrez[i][1]==symb[mrez[i][0]].repeat(2).toLowerCase())

{

rez+=m[mrez[i][0]].toLowerCase();

}

}

}

}

return rez;

};

class KnopkaFilter extends Component

{

mymethod(e)

{

$('.oshybka').remove();

};

render()

{

return(

<Button variant="contained" onClick={this.mymethod}>Убрать строки с ошибками</Button>

);

};

}

class KnopkaSort extends Component

{

mymethod(e)

{

var m= document.getElementsByClassName("zapis");

m = Array.prototype.slice.call(m);

m.sort(function(a, b){return a.innerHTML.localeCompare(b.innerHTML);});

var t=document.getElementById("rez\_div");

for(var i in m)

{

t.appendChild(m[i]);

}

};

render()

{

return(

<Button variant="contained" onClick={this.mymethod}>Сортировка↓</Button>

);

};

}

class KnopkaSortObr extends Component

{

mymethod(e)

{

var m= document.getElementsByClassName("zapis");

m = Array.prototype.slice.call(m);

m.sort(function(a, b){return -a.innerHTML.localeCompare(b.innerHTML);});

var t=document.getElementById("rez\_div");

for(var i in m)

{

t.appendChild(m[i]);

}

};

render()

{

return(

<Button variant="contained" onClick={this.mymethod}>↑</Button>

);

};

}

class KnopkaSpravka extends Component

{

mymethod(e)

{

var t = document.getElementById("rez\_div");

t.innerHTML="Пример имени: Михаил Владимирович Петров<br>";

t.innerHTML+="Пример шаблона: ИИ Оо Ф.<br>";

t.innerHTML+="Пример результата: МИХАИЛ Владимирович П.<br>";

t.innerHTML+="<b>Имя:</b><br>";

t.innerHTML+="И -> М<br>";

t.innerHTML+="и -> м<br>";

t.innerHTML+="Ии -> Михаил<br>";

t.innerHTML+="ИИ -> МИХАИЛ<br>";

t.innerHTML+="ии -> михаил<br>";

t.innerHTML+="<b>Аналогично: Ф - фамилия, О - отчество, П - пол, Н - номер (п/п)</b><br>";

t.innerHTML+="Для предотвращения замены символа, экранируйте его:<br><b>\\Ф -> Ф</b><br>";

}

render()

{

return(

<Button variant="contained" onClick={this.mymethod}>Справка</Button>

);

}

}

class Knopka extends Component

{

constructor(props,a)

{

super(props);

props.a.push(this);

this.m=props.a;

this.state = {txt:"Обработать"};

this.mymethod=this.mymethod.bind(this);

}

mymethod(e)

{

function tf(t)

{

if(document.getElementById("shablon\_vvod").value=="")

{

return;

}

if(document.getElementById("txt\_vvod").value=="")

{

return;

}

return function(iof)

{

document.getElementById("rez\_div").innerHTML="";

var m=JSON.parse(iof);

var sh=document.getElementById("shablon\_vvod").value;

for(var i in m)

{

iof=m[i];

document.getElementById("rez\_div").appendChild(document.createElement("div"));

var prav=iof["Правильность"];

delete iof["Правильность"];

document.getElementById("rez\_div").children[document.getElementById("rez\_div").children.length-1].innerHTML=shablon\_f(sh,iof);

if(!prav)

{

document.getElementById("rez\_div").children[document.getElementById("rez\_div").children.length-1].className="ispravleno";

}

if(document.getElementById("rez\_div").children[document.getElementById("rez\_div").children.length-1].innerHTML.trim().length==0)

{

document.getElementById("rez\_div").children[document.getElementById("rez\_div").children.length-1].innerHTML="! ("+document.getElementById("txt\_vvod").value.split("\n")[i]+")";

}

if((iof["Пол"]=="МЖ")||(iof["Пол"]=="Несоответствие"))

{

document.getElementById("rez\_div").children[document.getElementById("rez\_div").children.length-1].className="oshybka";

document.getElementById("rez\_div").children[document.getElementById("rez\_div").children.length-1].innerHTML="! ("+document.getElementById("txt\_vvod").value.split("\n")[i]+")";

}

document.getElementById("rez\_div").children[document.getElementById("rez\_div").children.length-1].classList.add("zapis");

}

}

}

const otvet=tf(this);

document.getElementById("rez\_div").innerHTML="";

$.ajax

(

{

type:"POST",

url: "http://localhost:8765/",

data: {"str":document.getElementById("txt\_vvod").value.split("\n")},

success:otvet

}

)

}

render()

{

return(

<Button variant="contained" onClick={this.mymethod}>{this.state.txt}</Button>

);

};

}

class Vvod extends Component

{

render()

{

const { classes } = this.props;

return(

<TextField label="Шаблон" variant="outlined" placeholder="Ии Оо Ф." id="shablon\_vvod" />

);

};

}

class TextVvod extends Component

{

render()

{

return(

<TextField label="Исходные имена" variant="outlined" multiline rowsMax="10" placeholder="Иванов Иван Иванович" id="txt\_vvod" />

);

};

}

class Rezultat extends Component

{

constructor(props,a)

{

super(props);

props.a.push(this);

this.state = {txt:""};

}

render()

{

return(

<div id="rez\_div" >{this.state.txt}</div>

);

};

}

class App extends Component

{

b=[];

render()

{

const { classes } = this.props;

return(

<div className="App">

<TextVvod />

<br />

<Vvod />

<Knopka a={this.b} /> <KnopkaSpravka />

<br />

<KnopkaFilter />

<KnopkaSort />

<KnopkaSortObr />

<br />

<Rezultat a={this.b} />

</div>

);

};

}

export default App;

**Файл /src/App.css:**

.App {

text-align: center;

margin: 10px;

}

.App-logo {

animation: App-logo-spin infinite 20s linear;

height: 40vmin;

pointer-events: none;

}

.App-header {

background-color: #282c34;

min-height: 100vh;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

font-size: calc(10px + 2vmin);

color: white;

}

.App-link {

color: #61dafb;

}

@keyframes App-logo-spin {

from {

transform: rotate(0deg);

}

to {

transform: rotate(360deg);

}

}

.ispravleno

{

background-color:yellow;

}

.oshybka

{

background-color:red;

}

**Файл /public/manifest.json:**

{

"short\_name": "Нормализация ФИО",

"name": "Нормализация ФИО",

"icons": [

{

"src": "https://reactjs.com/favicon.ico",

"sizes": "64x64 32x32 24x24 16x16",

"type": "image/x-icon"

}

],

"start\_url": ".",

"display": "standalone",

"theme\_color": "#000000",

"background\_color": "#ffffff"

}

1. Библиотека для нормализации ФИО, автор Анна Бучнева, источник: https://github.com/AnnaBuchneva/Algorithm\_FIO [↑](#footnote-ref-1)