

# Аннотация

Данная работа направлена на автоматизацию анализа информации о товарных знаках в международной системе товарных знаков Madrid Monitor. Целью работы является реализация веб-приложения, которое должно содержать составленные в ходе работы алгоритмы для автоматического определения актуального статуса защиты товарных знаков и актуального списка одобренных для защиты классов в заданной стране. В процессе исследования предметной области были изучены документы, регулирующие функционирование Мадридской системы, собраны статистические данные, отражающие определённые нюансы взаимодействия России с другими странами-резидентами Мадридской системы, придуманы алгоритмы анализа необходимой информации. Для разработки интерфейса приложения предпочтение было отдано популярному стеку технологий HTML, CSS, javascript, а при разработки серверной части были задействованы node.js и express. Серверная часть приложения получает клиентские запросы, при их обработке собирает и анализирует информацию из удалённого веб-сервиса Madrid Monitor и выводит данные на экран в удобной для пользователя форме.

# Abstract

This work aims to automate the analysis of trademark information in the Madrid Monitor international trademark system. The purpose of the work is to implement a web application that can contain algorithms drawn up during the work for determining a state of protection of a trademark and an up-to-date list of classes of a trademark approved for protection in a given country. In the process of researching the subject area, the documents regulating the functioning of the Madrid system were studied, statistical data, that determine the nuances of Russia's behavior with other resident countries of the Madrid system, were collected, and algorithms for analyzing the necessary information were constructed. For the development of the application, preference was given to the popular stack of technologies HTML, CSS, JavaScript, while node.js and express were involved in the development of the server side. The server side of the application receives client requests, during their processing, collects and analyses information from the Madrid Monitor remote web service and displays the data on the screen in a user-friendly form.

Содержание

[Введение 7](#__RefHeading___Toc9396_227585568)

[1. Обзор проблематики и постановка задачи 8](#__RefHeading___Toc9398_227585568)

[1.1 Обзор проблематики 8](#__RefHeading___Toc9400_227585568)

[1.2 Постановка задачи 9](#__RefHeading___Toc9402_227585568)

[2. Этапы и методы выполнения работы 10](#__RefHeading___Toc9404_227585568)

[2.1. Изучение документов 10](#__RefHeading___Toc9406_227585568)

[2.2. Дерево статусов заявок и решений стран по поводу защиты товарных знаков 11](#__RefHeading___Toc9408_227585568)

[2.3. Количество заявок, поданных из РФ и в РФ 13](#__RefHeading___Toc9410_227585568)

[2.4. Разработка веб-сервиса 15](#__RefHeading___Toc9412_227585568)

[3. Обзор технологий 25](#__RefHeading___Toc9414_227585568)

[3.1 Платформы и языки программирования 25](#__RefHeading___Toc9416_227585568)

[3.2 Веб-драйвер, используемый для сбора статистических данных 26](#__RefHeading___Toc9418_227585568)

[3.3 Программное обеспечение (ПО), использованное при разработке серверной части веб-приложения 26](#__RefHeading___Toc9420_227585568)

[3.4 Языки разметки и стилей, использованные в веб-сервисе 31](#__RefHeading___Toc9422_227585568)

[4. Структура веб-приложения 32](#__RefHeading___Toc5838_872103484)

[Серверная часть: 32](#__RefHeading___Toc5840_872103484)

[Клиентская часть: 32](#__RefHeading___Toc5842_872103484)

[Заключение 34](#__RefHeading___Toc9424_227585568)

[Список использованных источников 35](#__RefHeading___Toc9426_227585568)

[Приложение 1 36](#__RefHeading___Toc9428_227585568)

[Приложение 2 39](#__RefHeading___Toc9430_227585568)

# Введение

Проблема защиты интеллектуальной собственности является очень актуальной для большинства компаний, которые производят товары или предоставляют услуги на рынке. В подавляющем большинстве стран существуют специальные органы, занимающиеся контролем соблюдения прав в этой сфере. Для поощрения и защиты интеллектуальной собственности во всем мире путем сотрудничества со странами, а также с международными организациями, во второй половине двадцатого века была подписана конвенция об учреждении Всемирной организации интеллектуальной собственности (далее — ВОИС). Одним из ряда пунктов сферы деятельности ВОИС является осуществление регистрации товарных знаков для защиты на территории стран, входящих в Мадридскую систему. Мадридская система [1] — это удобный механизм регистрации товарных знаков и управления ими по всему миру. Сайт ВОИС предоставляет платформу, позволяющую отслеживать все изменения в судьбе товарных знаков, зарегистрированных по процедуре Мадридской системы. Данный сервис называется Madrid Monitor. Патентные организации заинтересованы в услугах, которые упрощают навигацию по базе данных Madrid Monitor.

Данная работа направлена на упрощение поиска информации и на автоматизацию анализа статуса защиты товарного знака в любой стране на основе информации, получаемой из сервиса Madrid Monitor.

# 1. Обзор проблематики и постановка задачи

## 1.1 Обзор проблематики

Для автоматического анализа статуса защиты заданного товарного знака в заданной стране необходимо иметь доступ к информации, предоставляемой сервисом Madrid Monitor. Существует несколько различных способов получения такой информации.

Самым простым способом для разработки веб-приложения, автоматически анализирующего данные о товарном знаке, является получение платного доступа к базе данных ВОИС. Прямой доступ к базе данных обеспечивает разработчику быстрый способ получения интересующей его информации для последующей работы с ней. Однако некоторые организации, заинтересованные в таком веб-приложении, не готовы оплачивать данную услугу, предоставляемую ВОИС. В таком случае приходится придумывать бесплатные, но менее быстрые и надёжные способы получения данных из сервиса Madrid Monitor. Данная работа описывает подход получения нужных данных без оплаты доступа к базе данных ВОИС.

До определённого времени в Madrid Monitor была возможность получить информацию о заданном товарном знаке по универсальной ссылке на XML-разметку, в которую оставалось вставить лишь нужный номер международной регистрации товарного знака. Однако сейчас доступ по такой ссылке ограничен, и для каждого запроса через интерфейс сервиса Madrid Monitor создаётся страница с XML-разметкой по уникальной временной ссылке. XML-разметка страницы заданного товарного знака в системе Madrid Monitor содержит достаточно структурированную информацию, работать с которой, как мне кажется, более удобно, чем с HTML-разметкой такой же страницы, но программным путём на стороне сервера мне не удалось получить уникальную ссылку, генерируемую системой.

В результате был выбран вариант сбора и структурирования информации (далее – парсинга) из HTML страницы заданного товарного знака. В сервисе Madrid Monitor HTML-разметка не очень удобно структурирована для парсинга, поэтому при разработке веб-приложения пришлось использовать много регулярных выражений. Ещё одной проблемой является то, что для получения всей необходимой информации из системы по HTML, необходимо нажать определённую кнопку на странице заявки на регистрацию. Сымитировать нажатие кнопки на удаленной веб-странице можно с помощью различных веб-драйверов. Недостатком использования этих драйверов является то, что они значительно замедляют скорость выполнения программы по сравнению с обычным парсингом веб-страниц без имитации нажатия кнопок.

Важной задачей при разработке веб-приложения было придумать алгоритм автоматического анализа статуса регистрации заданного пользователем товарного знака в любой заданной стране.

## 1.2 Постановка задачи

Разработать веб-сервис, способный автоматически определять статус международной охраны товарного знака, заданного пользователем по номеру регистрации в международной системе товарных знаков Madrid Monitor. По возможности реализовать атвоматическое определение одобренного для защиты списка классов заданной пользователем страной. Веб-сервис должен обладать интуитивно понятным интерфейсом. Задача поставлена Федеральным институтом промышленной собственности (ФИПС).

# 2. Этапы и методы выполнения работы

Для выполнения поставленной задачи необходимо было выполнить множество различных шагов от ознакомления с работой Мадридской системы и документами, регулирующими её деятельность, до изучения и применения множества технологий, позволяющих реализовать веб-приложение.

## 2.1. Изучение документов

Мадридская система регулируется двумя основными документами: Мадридским соглашением о международной регистрации знаков и Протоколом к Мадридскому соглашению о международной регистрации знаков.

Эти документы имеют множество противоречий в своих статьях. Различные страны подчиняются либо соглашению, либо протоколу, однако в современных реалиях большинство участников Мадридского союза присоединяются к нему по Протоколу к Мадридскому соглашению о международной регистрации знаков. Перечисленные документы дополняются Общей инструкцией к Мадридскому соглашению и Протоколу у этому соглашению.

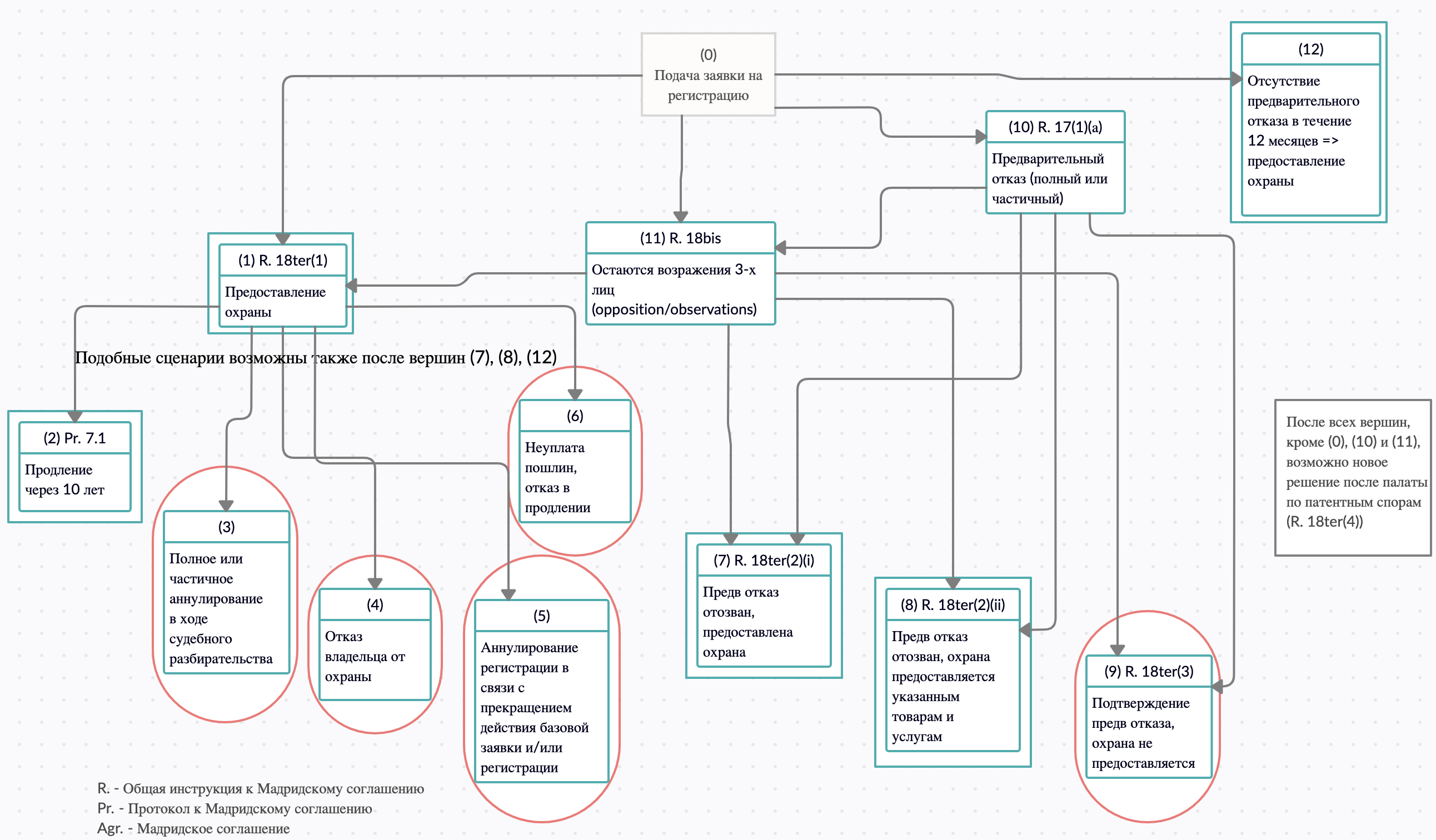
Документы необходимо было изучить для понимания того, что происходит с международной регистрацией товарного знака в Мадридской системе на разных этапах изменения её статуса, а также для продумывания алгоритма автоматического анализа статуса заявки в выбранной стране.

## 2.2. Дерево статусов заявок и решений стран по поводу защиты товарных знаков

На основе изученных документов путём эмпирического исследования было составлено дерево решений (Рис. 1) стран, в которые подаются заявки на регистрацию защиты товарного знака. Опираясь на это дерево, я составлял алгоритм определения актуального статуса защиты товарного знака.

На картинке с деревом решений (Рис. 1) можно увидеть, как связаны и в каком порядке могут приниматься решения стран по защите какого-либо товарного знака. Зелёной и красной дополнительными рамками обозначаются активный и неактивный статус защиты товарного знака соответственно. На определённых узлах дерева статус заявки может быть не определён, тогда дополнительной рамки на этих узлах нет. Активный статус защиты товарного знака означает, что выбранная страна предоставляет защиту на своей территории хотя бы для одного заявленного при регистрации товарного знака в системе класса. Неактивный статус значит, что ни одному классу защита не предоставляется.

Классы определяются согласно Международной классификации товаров и услуг (подробно описано в статье [2]), принятой в рамках Ниццкого соглашения о классификации товаров и услуг для регистрации знаков. Заявленные для защиты классы включают в себя товары и услуги, которым может предоставляться защита на территории стран-резидентов Мадридской системы.

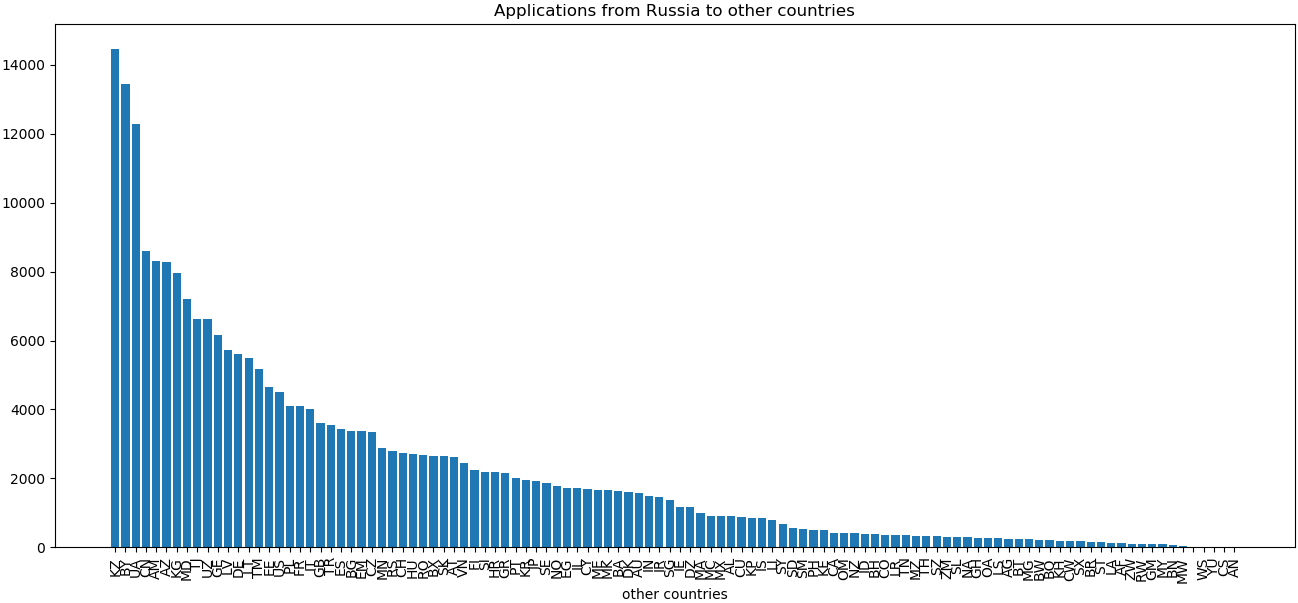
Рис. 1. Дерево возможных статусов заявок и возможных решений стран, в которые подаются заявки на регистрацию, по поводу защиты товарных знаков

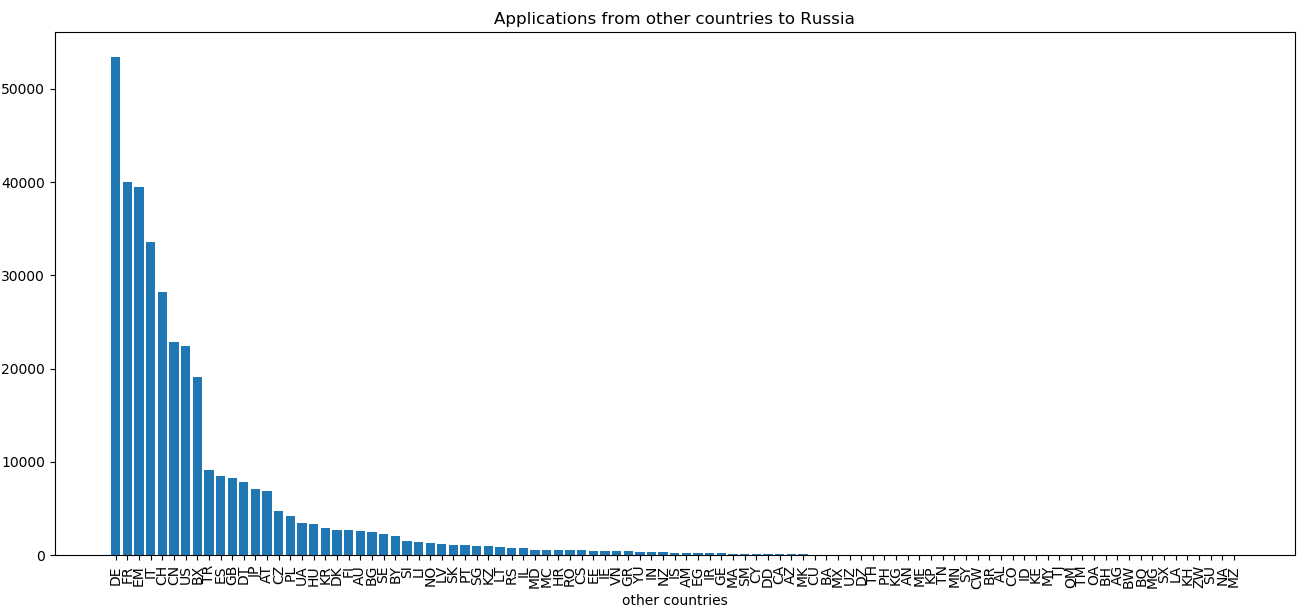
## 2.3. Количество заявок, поданных из РФ и в РФ

В рамках исследования предметной области программно, путём имитации прокликивания нескольких тысяч страниц в браузере (при помощи Selenium WebDriver с использованием драйвера Chrome) в системе Madrid Monitor и сбора информации с каждой страницы, была составлена статистика в виде гистограмм (Рис. 2 и Рис. 3). Столбцы гистограмм отражают, сколько заявок было подано из России в другие страны-резиденты Мадридской системы (Рис. 2) и сколько заявок было подано из других стран в Россию (Рис. 3).

Результаты получены 6 марта 2021 г. и не являются актуальными в данный момент. Однако по ним можно понять, в каких странах российские компании чаще всего пытаются защитить свои товарные знаки, и компании каких стран с большим желанием пытаются защитить свои товарные знаки на территории РФ.

Также подобная статистика и метод её получения могут быть полезны в будущем при исследовании объёма защиты (среднее отношения количества одобренных классов для защиты к количеству заявленных классов) классов российских компаний за рубежом в конкретных странах в сравнении друг с другом и объёма защиты классов зарубежных компаний в России в сравнении друг с другом.

 Рис. 2. Статистические данные, отражающие количество заявок, поданных из России в другие страны-резиденты Мадридской системы

 Рис. 3. Статистические данные, отражающие количество заявок, поданных из других стран-резидентов Мадридской системы в Россию

## 2.4. Разработка веб-сервиса

Разработка веб-приложения состояла из следующих шагов:

1) **Получение HTML-разметки страницы товарного знака из сервиса Madrid Monitor с использованием PhantomJS.**

PhantomJS - это «безголовый» (выполнение отображается не на дисплее браузера, а в командной строке) веб-браузер, работающий с JavaScript. Он работает в Windows, macOS, Linux и FreeBSD. Предпочтение было отдано ему, поскольку для получения всей необходимой информации из сервиса Madrid Monitor, на странице товарного знака нужно нажимать на кнопку с обозначением определённой страны и получать изменённую после нажатия кнопки разметку. То, что он является безголовым, является преимуществом перед распространённым веб-драйвером Selenium по скорости выполнения. Также на выбор использования именно PhantomJS повлияло то, что на его основе написана библиотека phantomjs-pool, устанавливаемая на node js с помощью пакетного менеджера NPM (Node Package Manager). Данная библиотека в проекте необходима для того, чтобы обеспечить возможность параллельного выполнения работы PhantomJS на сервере при получении нескольких запросов от пользователей на поиск в одно и то же время.

2) **Парсинг HTML-разметки с использованием регулярных выражений для получения данных о товарном знаке.**

Полученная HTML-разметка парсится с помощью пакета node-html-parser. Данные о товарном знаке, такие как наименование товарного знака и его регистрационный номер, дата регистрации товарного знака, язык регистрации товарного знака, детали владельца товарного знака, зарегистрированный список классов, события, связанные с товарным знаком в выбранной стране, статус заявки в выбранной стране, события, связанные с изменениями заявленных классов в выбранной стране, актуальный список классов в выбранной стране, достаются из HTML с помощью вышеупомянутого пакета. Во многих случаях для получения нужных данных необходимо использовать регулярные выражения. Для случаев со статусом заявки в выбранной стране и актуальным списком классов в выбранной стране далее подробно будут описаны алгоритмы их определения.

3) **Определение статуса защиты товарного знака на основе полученных данных о товарном знаке и придуманного алгоритма.**

Алгоритм определения статуса защиты товарного знака строится на знаниях, полученных после изучения документов, связанных с Мадридской системой, и на дереве решений стран по поводу защиты товарных знаков и статусов заявок. В дереве решений обозначены основные значимые события, которые влияют на статус защиты товарного знака, однако на практике таких событий больше, чем узлов в этом дереве. Чтобы получить полный перечень таких событий, был выполнен запрос к локальной базе данных ФИПСа. База данных выдала 64 события, из которых 46 оказались влияющими на статус защиты товарного знака.

На основе всех этих событий был составлен двумерный массив с самими событиями и обозначениями (далее – сигналами), сигнализирующими их влияние на изменение статуса защиты товарного знака, и вынесен в отдельный конфигурационный файл. Определены 4 сигнала: «a» – сигнал, обозначающий, что товарному знаку предоставляется защита на территории выбранной страны; «i» – сигнал, обозначающий, что товарному знаку не предоставляется защита на территории выбранной страны; «n» – сигнал, обозначающий, что событие, к которому он относится, не влияет на статус защиты товарного знака на территории выбранной страны; «r» – сигнал, обозначающий, что защита предоставляется либо не предоставляется товарному знаку на территории выбранной страны в зависимости от определённого обстоятельства. Также во время обработки клиентского запроса на поиск информации о товарном знаке каждый раз формируется массив событий, связанных именно с заданным товарным знаком в хронологическом порядке. Этот массив берётся из HTML-разметки страницы товарного знака. Осуществляется проход по данному массиву с конца к началу. Каждый элемент этого массива соответствует одному элементу массива из конфигурационного файла. При проходе по событиям из этого массива для каждого элемента осуществляется поиск такого же события в массиве из конфигурационного файла (методом indexOf(). Метод возвращает индекс первого вхождения подстроки в строку. Если он вернул не -1, значит, событие из массива из конфигурационного файла соответствует текущему событию, связанному с выбранным товарным знаком) и анализ сигнала, относящегося в подходящему событию. Если событию соответствует сигнал «n», программа пропускает данное событие, относящееся к товарному знаку, и переходит к следующему; если событию соответствует сигнал «a», программа прекращает проход по массиву событий, связанному с товарным знаком и присваивает товарному знаку статус «Active» (защита товарного знака осуществляется); если событию соответствует сигнал «i», программа прекращает проход по массиву событий, связанному с товарным знаком и присваивает товарному знаку статус «Inactive» (защита товарного знака не осуществляется); если событию соответствует сигнал «r», программа проверяет наличие определённого дочернего элемента этого события в HTML-разметке и в зависимости от наличия этого элемента присваивает товарному знаку статус «Active» либо «Inactive». В конце работы алгоритма, если товарному знаку присвоен статус «Active», производится дополнительная проверка, определяющая, является ли срок защиты товарного знака истёкшим в выбранной стране или не является.

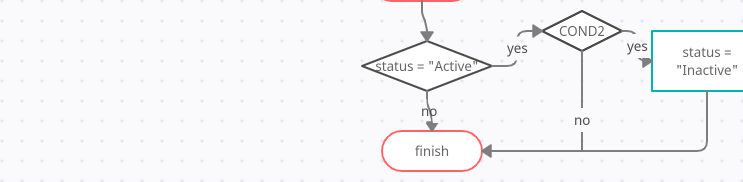
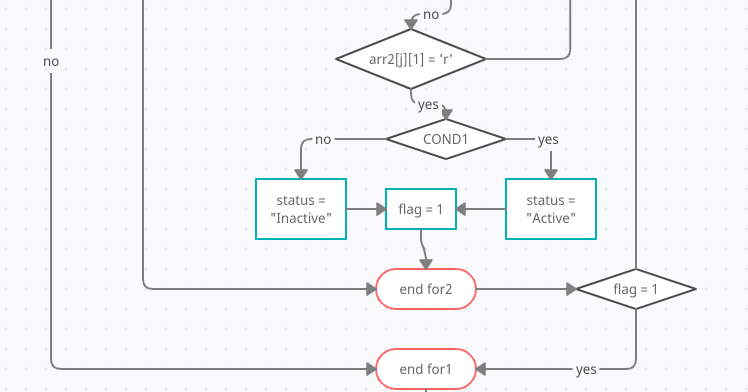
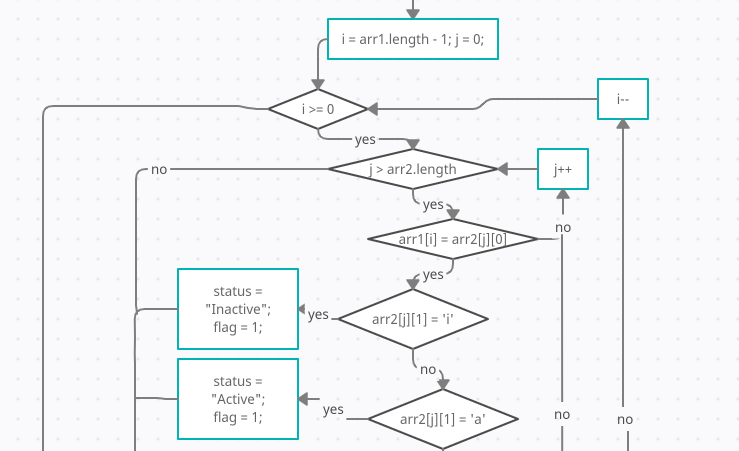
**Блок-схема (Рис. 4), в общих чертах (за исключением некоторых надстроек) описывающая принцип работы алгоритма определения статуса защиты товарного знака:**

На блок-схеме задействованы переменные status и flag. Их начальные значения: status = «not found»; flag = 0.

Также на ней используются значения элементов массивов arr1 и arr2.

arr1 – одномерный массив, элементы которого являются событиями, относящимися к указанному пользователем товарному знаку. arr2 – двумерный массив, элементы которого содержат по 2 элемента с индексами 0 и 1. Индексом 0 пронумерованы элементы, являющиеся событиями, собранными из локальной базы данных ФИПСа. Индексом 1 пронумерованы элементы, определяющие влияние соответствующего им события (под индексом 0) на статус защиты товарного знака (сигналы).

Блоки под названиями «COND1» и «COND2» являются условиями, где «COND1» проверяет, имеется ли в HTML-разметке среди дочерних элементов выбранного элемента событие «The refusal period has expired and no notification of provisional refusal has been recorded (application of Rule 5 preserved)», «COND2» проверяет, является ли регистрация товарного знака просроченной или не является.

Рис. 4. Блок-схема алгоритма определения статуса защиты товарного знака

4) **Определение актуального списка классов, охраняемых на территории заданной страны, на основе регулярных выражений.**

Для определения актуального списка классов первым делом необходимо было понять, какие формулировки влияют на изменение этого списка. Для того, чтобы это понять, был выполнен ещё один запрос к локальной базе данных ФИПС, с помощью которого было получено огромное количество уникальных формулировок, влияющих на изменение списка классов. Отсортировав список этих формулировок и изучив его, были выделены основные формулировки, на основе которых можно строить регулярные выражения и находить события, влияющие на изменение изначального списка классов для защиты в выбранной стране. На основе регулярных выражений и метода indexOf() среди всех событий, связанных с товарным знаком в выбранной стране, выбираются только те события, которые сигнализируют об изменении списка заявленных классов. Выбранные события записываются в массив. Элементы этого массива также анализируются с помощью регулярных выражений.

После анализа каждого из элементов выбираются классы, товары или услуги, которые нужно удалить, изменить или добавить в исходный массив заявленных при регистрации классов.

Исходный массив заявленных при регистрации товарного знака в Мадридской системе классов формируется при каждом запросе пользователя на поиск информации о товарном знаке.

При таком подходе хорошо определяются события, влияющие на изменение списка классов, но встречаются неформализованные случаи для изменения изначального списка классов. В таких случаях пользователю сообщается, что определить актуальный список классов невозможно и предлагается самостоятельно проанализировать события, влияющие на изменение изначального списка классов. В любом случае, даже если определить актуальный список классов не удаётся, предоставление пользователю списка событий, влияющих на изменение списка классов, значительно сокращает время определения пользователем конечного актуального списка классов, охраняемых на территории заданной страны.

**Пошаговое описание алгоритма, определяющего актуальный перечень товаров и услуг, на которые распространяется защита в заданной стране:**

а) Проход по массиву событий, связанных с указанным товарным знаком.

б) Добавление в новый массив событий, связанных с изменением заявленного перечня классов, товаров и услуг, только тех событий которые содержат хотя бы одну из следующих подстрок: «goods», «services», «for which protection of the mark is granted», «specification»; или которые не равны null при наложении на них хотя бы одного из следующих регулярных выражений: «/list(.|\n)\*limit/», «/limit(.|\n)\*list/», «/list(.|\n)\*delete/», «/delete(.|\n)\*list/».

в) Применение к каждому элементу, добавленному в новый массив, функции actualClassListFunc(), которая выполняет следующее:

найдя в строке события (элементе нового массива) подстроку «Delete from list», удаляет из массива с изначально зарегистрированным перечнем товаров и услуг все перечисленные в событии товары и услуги;

найдя подстроку «List limited to», заменяет в массиве с зарегистрированным перечнем все товары и услуги на перечисленные в событии только у тех классов, которые перечислены в этом событии;

найдя подстроку «for which protection of the mark is granted», заменяет данные массива с зарегистрированным перечнем на калссы, товары и услуги, указанные в событии;

наложив на строку с событием следующие регулярные выражения : «/refus(.|\n)\*for all(.|\n)\*in class(.|\n)\*?[;|\.]/», «/invalid(.|\n)\*for all(.|\n)\*in class(.|\n)\*?[;|\.]/», «/class(.|\n)\*cancell/», определяет, равно наложение null или нет, и если не равно, удаляет из зарегистрированного перечня все классы, перечисленные в событии.

5) **Разработка интерфейса веб-приложения, в котором клиент может вводить номер международной регистрации товарного знака и выбирать страну, на территории которой проверяется активность статуса защиты товарного знака и определяется актуальный список классов.**

Интерфейс веб-приложения разработан с помощью популярного стэка технологий: HTML, CSS, javascript. Для удобного взаимодействия с сервером и удобного применения внутри HTML-разметки языка программирования javascript использованы EJS (Embedded JavaScript templating) шаблоны. Весь интерфейс состоит из 3 файлов с расширением .ejs, которые воспроизводятся пользователю сервером при определённых запросах пользователя. Все 3 файла содержат HTML-разметку веб-приложения с частичными вставками javascript-кода. Стили применяются ко всем трём шаблонам с помощью одного публичного файла с расширением .css.

6) **Встраивание скрипта по поиску и анализу информации, полученной из системы Madrid Monitor, в серверную архитектуру, принимающую и обрабатывающую пользовательские запросы и возвращающую результаты работы скрипта пользователю через интерфейс веб-сервиса.**

Когда готов скрипт, получающий и анализирующий всю информацию из системы Madrid Monitor, нужно уметь применять его в тот момент, когда пользователь даст для этого команду.

Такая команда осуществляется пользователем через интерфейс веб-приложения при нажатии на кнопку «Поиск» после заполнения пользователем номера заявки и выбора нужной страны, в которой нужно проверить статус защиты товарного знака и определить актуальный список классов.

После нажатия кнопки на сервер отправляется get-запрос, обрабатываемый с помощью фреймворка express. Обработка такого запроса заключается в использовании ранее описанного скрипта, в который пользователем подаются только номер заявки и обозначение страны. После того, как скрипт отрабатывает, программа пакует все данные, которые нужно вернуть пользователю в формат JSON и посылает такой пакет данных клиенту вместе с воспроизведённым на экран нужным шаблоном (в конечном итоге search.ejs). Таким образом, пользователь видит на экране нужную ему информацию в виде понятной ему веб-страницы с заполненными текстовыми блоками.

До того, как пользователь нажмёт кнопку «Поиск», он вводит сначала регистрационный номер заявки, нажимает на кнопку «Подтвердить», нажатие которой также является запросом к серверу. На этом этапе сервер проверяет корректность введённого номера заявки и наличие такого товарного знака в системе (с помощью ), находит все страны, в которые данная заявка была подана для защиты соответствующего товарного знака, пакует обозначения найденных стран в JSON формат и возвращает пользователю с воспроизведённым шаблоном designations.ejs. После выполнения таких действий пользователь может выбрать из выпадающего списка нужную ему страну и нажать на кнопку «Поиск».

Все поля для заполнения пользователем реализованы в виде формы, данные которой отправляются на сервер вместе с get-запросом.

С использованием express сервер работает через порт, находящийся в переменной окружения PORT, если там что-то находится, или через 3000 порт.

Стоит заметить, что пользователь может обращаться к серверу не только через интерфейс веб-приложения, но и по прямой ссылке, дописав к домену нужное имя файла запрашиваемой страницы и необходимые параметры. Например, чтобы перейти к заполнению номера заявки, требуется ссылка на веб-сервис лишь с заполненными протоколом и именем домена. Для прямого перехода к выбору обозначения страны с уже заполненным номером заявки необходимо дописать после имени домена следующее: /designations?number=<номер заявки> (например, http://localhost:3000/designations?number=739890). Для выполнения поиска информации о товарном знаке с определением статуса заявки и актуального списка классов, после имени домена нужно дописать следующее: /search?number=<номер заявки>&chosen\_desig=<обозначение страны> (например, http://localhost:3000/search?number=739890&chosen\_desig=RU).

В данный момент сервер развёрнут на бесплатном хостинге Heroku, а значит, любой пользователь может воспользоваться веб-приложением по URL <https://marid-info.herokuapp.com/>.

# 3. Обзор технологий

В этой главе будет осуществлён обзор технологий, которые применялись на всех этапах разработки веб-сервиса.

## 3.1 Платформы и языки программирования

**Среда разработки: Visual Studio Code**

Visual Studio Code – это текстовый редактор исходного кода, который содержит множество вспомогательных плагинов и дополнительного функционала, помогающего разработчику решать задачи сборки проекта, его компиляции, отладки и не только.

**Платформа для разработки веб-приложения: node js**

Node js – это платформа, транслирующая JavaScript-код в машинный код. Платформа удобна для разработки веб-сервера. Содержит пакетный менеджер NPM, позволяющий быстро и легко устанавливать нужные программные модули в приложение.

**Язык программирования, используемый для разработки веб-приложения: JavaScript**

JavaScript [3] — это слабо типизированный язык программирования для работы в браузере. Он является одним из самых распространённых языков программирования. Платформа Node js позволяет использовать JavaScript вне браузеров, благодаря чему он стал также очень популярен среди разработчиков ПО.

**Язык программирования, используемый для составления статистики: Python**

Python [4] — это высокоуровневый язык программирования общего назначения, который используется в том числе и для разработки веб-приложений.

## 3.2 Веб-драйвер, используемый для сбора статистических данных

Selenium WebDriver (драйвер браузера) [5] – это программная библиотека, которая не имеет пользовательского интерфейса и позволяет различным программам управлять поведением браузера и получать из него какую-либо информацию.

Веб-драйвер воспроизводит на экране работу браузера (в данной работе был выбран драйвер браузера Chrome) под автоматизированным управлением запущенного приложения, что позволяет следить за работой программы в режиме реального времени.

Selenium WebDriver не поддерживает корректную параллельную работу нескольких созданных объектов веб-драйвера, поэтому он был использован только при сборе статистики одним объектом. Для разработки веб-сервиса был использован «безголовый» веб-браузер PhantomJS, предоставляющий возможность работы в нескольких потоках.

## **3.3 Программное обеспечение (ПО), использованное при разработке серверной части веб-приложения**

Вся серверная часть веб-приложения описана в файле index.js, который получает определённый функционал из подключённых к нему файлов config.js и worker.js.

**Парсеры: Osmosis, node-html-parser, body-parser**

Парсер – это ПО, способное выделять нужную пользователю информацию из массива данных.

Почти вся информация в Интернете существует в виде HTML-страниц. Информация на этих страницах структурирована в виде элементов HTML-разметки. Элементы структурированы по вертикальной иерархической древовидной структуре (родительские и дочерние). Такая структура называется Document Object Model (DOM). Благодаря появлению этой структуры стал удобен процесс извлечения информации из сети, который называется «scraping». «scraping» веб-страниц используется в данном веб-приложении для автоматизации повторяющихся задач сбора информации.

Osmosis – веб-парсер HTML или XML разметок, который, в отличие от множества распространённых парсеров, способен получать полный доступ к динамической HTML-странице со встроенным javascript-кодом. В веб-приложении он применяется при обработке первого запроса пользователя, когда он ввёл номер заявки на регистрацию и подтвердил свой выбор, для проверки существования такого товарного знака в системе Madrid Monitor и для возврата пользователю списка стран, в которые была подана соответствующая регистрация.

Node-html-parser – очень быстрый парсер HTML, приоритетом которого является производительность. Преобразует HTML-разметку страницы товарного знака из сервиса Madrid Monitor, полученную при помощи PhantomJS WebKit, в формат JSON. Далее в приложении используется для получения необходимой для анализа и возвращения пользователю информации из преобразованного из HTML в JSON объекта.

Body-parser – промежуточное ПО, анализирующее входящие от клиента на сервер запросы. В приложении используется промежуточное ПО, возвращаемое с помощью метода urlencoded. С его помощью парсятся параметры, передаваемые клиентом на сервер через форму GET-запроса.

**Javascript фреймворки** – коллекции библиотек кода JavaScript, которые предоставляют разработчикам предварительно написанный JS-код для решения различных задач программирования.

**Фреймворк, способный параллельно в фоновом режиме выполнять указанные пользователем автоматизированные действия в браузере:**

PhantomJS – это не связанный с браузером обособленный web-стек на базе движка WebKit, предназначенный для автоматизированной обработки веб-страниц . В главе 1 (Этапы и методы выполнения работы), пункте 1.4, уже перечислены основные преимущества данного ПО и его назначение в серверной части веб-приложения. В веб-приложении за функционирование данного движка отвечает исходный код, находящийся в файле worker.js.

Phantomjs-pool – библиотека, использующая движок PhantomJS и реализованная на основе порождающего паттерна проектирования под названием «объектный пул». Именно его реализация на основе этого паттерна позволяет серверу параллельно обрабатывать несколько клиентских GET-запросов на поиск и анализ информации о товарном знаке в системе Madrid Monitor. «Объектный пул» – это набор инициализированных и готовых к использованию объектов, которые не создаются, а берутся из пула, когда они требуются, и возвращаются в пул, когда они больше не нужны.

**Регулярные выражения:**

В данном веб-сервисе регулярные выражения играют значительную роль в работе алгоритмов автоматического анализа при анализе текстовых фрагментов, полученных из сервиса Madrid Monitor.

Регулярное выражение [6] – это объект, описывающий текстовый шаблон. Класс JavaScript RegExp определяет методы, которые используют регулярные выражения для выполнения мощных функций сопоставления с образцом, поиска и замены текста. Однако, чтобы эффективно использовать API RegExp, нужно уметь описывать шаблоны текста с помощью грамматики регулярных выражений, которая, по сути, представляет собой собственный мини-язык программирования.

**Фреймворк для обработки клиентских запросов: Express**

Express [7] – это быстрая, минималистичная веб-среда для Node.js, используемая в производственных средах по всему миру. Express – это прямая замена для основного модуля http. Самая большая разница между использованием экспресс-протокола и базового HTTP - это маршрутизация, потому что, в отличие от базового HTTP, экспресс-версия имеет встроенный маршрутизатор.

С помощью данного фреймворка настроена вся серверная архитектура. С помощью метода .listen() устанавливается порт, через который сервер получает клиентские запросы, с помощью методов .set() и .use() подключаются шаблоны и стили для всех шаблонов, с помощью метода .get() обрабатываются клиентские GET-запросы и воспроизводятся шаблоны с HTML-разметкой страниц приложения и переданной клиенту в JSON формате информацией, проанализированной внутри обработчиков клиентского запроса.

**Передача данных между сервером и клиентом**

Коммуникация клиента и сервера происходит по протоколу HyperText Transfer Protocol (HTTP). Такую коммуникацию обеспечивает вышеописанный фреймворк express. HTTP [8] – протокол, описывающий взаимодействие между клиентом и сервером, построенное на базе сообщений, называемых запрос (Request) и ответ (Response).

Тип HTTP-запроса указывает серверу на то, какое действие мы хотим произвести с ресурсом.

Основные типы запросов к серверу (их существует больше, чем 4 перечисленных):

GET — получение ресурса

POST — создание ресурса

PUT — обновление ресурса

DELETE — удаление ресурса

Сервер данного веб-приложения обрабатывает только GET-запросы пользователя, которые реализованы программно в виде форм в HTML-разметке. Форма заполняется через интерфейс приложения и отправляется в виде запроса клиентом на сервер. Сервер обрабатывает данные запросы и отправляет пользователю ответ, который генерирует обработчик запроса.

GET-запрос [9] – это метод передачи данных от клиента к серверу с целью получения информации, указанной с помощью конкретных GET-параметров. Это публичные данные, доступные при повторном просмотре ссылки в истории.

В ответе сервера содержатся EJS шаблоны с HTML-разметкой, которая воспроизводит интерфейс приложения, и некоторые данные, упакованные в формат JSON, которые добавляются в шаблон с помощью JavaScript-кода.

JSON – это текстовый формат, который хранит структурированную информацию и в основном используется для передачи данных между сервером и клиентом. Данные в нём хранятся в виде {ключ: значение}.

При переходе на главную страницу приложения сервер воспроизводит клиенту файл (шаблон) index.ejs, после заполнения номера заявки и его подтверждения – designations.ejs, после выбора нужного обозначения страны и нажатия на кнопку поиск – search.ejs. Все перечисленные шаблоны лежат в папке views. Папка views находится в корневой директории приложения.

EJS - это простой язык шаблонов, который позволяет создавать разметку HTML с помощью простого JavaScript.

## 3.4 Языки разметки и стилей, использованные в **веб-сервисе**

Описание структуры веб-страниц реализовано с помощью HTML-разметки. HTML [10] – это набор правил по разметке документов, использующийся для того, чтобы дать браузеру понять, как нужно отображать загруженный сайт. HTML состоит из тэгов – команд, которые преобразовываются в объекты в браузере пользователя. После указания, где какие элементы должны располагаться на веб-странице, нужно описать внешний вид веб-страницы. Осуществляется это с помощью таблицы стилей, которая называется Cascading Style Sheets (CSS). Благодаря CSS конфигурируется внешний вид страницы с её цветами, анимацией, отступами и так далее.

# 4. Структура веб-приложения

## Серверная часть:

**файл index.js**

Содержит основную реализацию серверной архитектуры с обработчиками GET-запросов, соединением с клиентом по HTTP-протоколу, алгоритмами получения и анализа информации о товарном знаке из сервиса Madrid Monitor.

**файл worker.js**

Содержит запрограммированную последовательность действий «безголового» веб-браузера PhantomJS, которая заканчивается тем, что в файле index.js принимается HTML-разметка страницы товарного знака в Madrid Monitor, переданная из этого файла.

**файл config.js**

Содержит код инициализации и экспорта в файл index.js двумерного массива со всеми возможными событиями, влияющими на статус защиты товарного знака, и сигналами («a», «i», «n», «r»), определяющими значимость передаваемых событий.

## Клиентская часть:

**файлы index.ejs, designations.ejs, search.ejs, находящиеся в папке views**

Все 3 файла являются шаблонами EJS и содержат HTML-разметку и JavaScript-код. index.ejs и designations.ejs среди HTML-кода содержат формы, заполняемые клиентом и передающиеся на сервер для обработки в качестве запросов. index.ejs формирует главную страницу с заполнением номера регистрации товарного знака, designations.ejs – страницу с выбором обозначения страны, search.ejs – страницу с итоговой информацией о товарном знаке.

**файл style.css, находящийся в папке public**

Содержит CSS-стили для трёх шаблонов EJS, то есть, задаёт стилизацию интерфейсу веб-приложения.

# Заключение

В ходе работы были изучены правовые механизмы регуляции Мадридской системы и рассмотрены современные методы и технологии веб-разработки. Был описан процесс создания веб-приложения, позволяющего проводить автоматический анализ регистрационных заявок на защиту товарных знаков в любой стране, на основе клиент-серверной архитектуры. Разработанный веб-сервис позволяет сэкономить время по поиску и анализу информации о товарном знаке в системе Madrid Monitor любым пользователям и организациям, заинтересованным в этом, так как находится в открытом доступе в сети интернет. В ходе работы над данным проектом открылись новые перспективы исследования предметной области и совершенствования функционала разработанного веб-приложения, например, добавление функционала, способного автоматически определять объём предоставления защиты товарным знакам одной страны другими странами-резидентами Мадридской системы.

# Список использованных источников

1. «Мадридская система – международная система товарных знаков». URL: <https://www.wipo.int/madrid/ru/> (дата обращения 16.05.2021)

2. «Что такое классы МКТУ и для чего они нужны?». URL: <https://zuykov.com/ru/about/articles/2016/08/05/chto-takoe-klassy-mktu-i-dlya-chego-oni-nuzhny/> (дата обращения 16.05.2021)

3. Flanagan D. JavaScript: The Definitive Guide. Seventh ed. Sebastopol: O’Reilly Media, Inc., 2020. P. 1-2.

4. «Язык программирования Python». URL: <https://web-creator.ru/articles/python/> (дата обращения 17.05.2021)

5. «Что такое Selenium WebDriver?». URL: <https://habr.com/ru/post/152971/> (дата обращения 17.05.2021)

6. Flanagan D. JavaScript: The Definitive Guide. Seventh ed. Sebastopol: O’Reilly Media, Inc., 2020. P. 281.

7. Guttman D. Fullstack Node.js. Fullctack.io. 2019. P. 20.

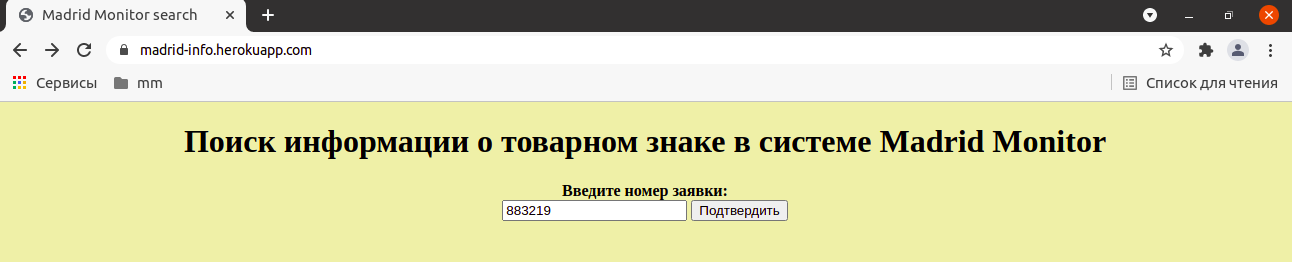
8. «Типы HTTP-запросов и философия REST». URL: <https://habr.com/ru/post/50147/> (дата обращения 17.05.2021)

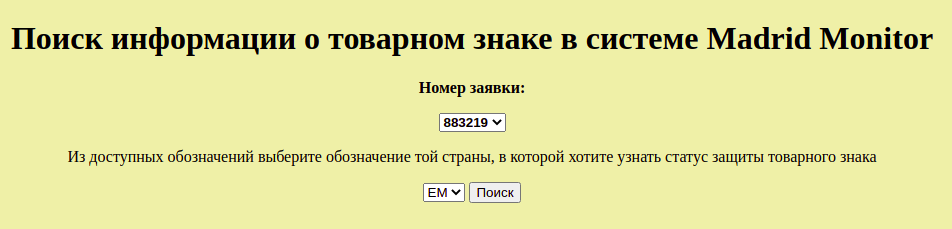
9. «Сколько GET-параметров может находиться в URL страницы». URL: <https://serpstat.com/ru/blog/skolko-get-parametrov-mozhet-nahoditsja-v-url-stranicy/> (дата обращения 17.05.2021)

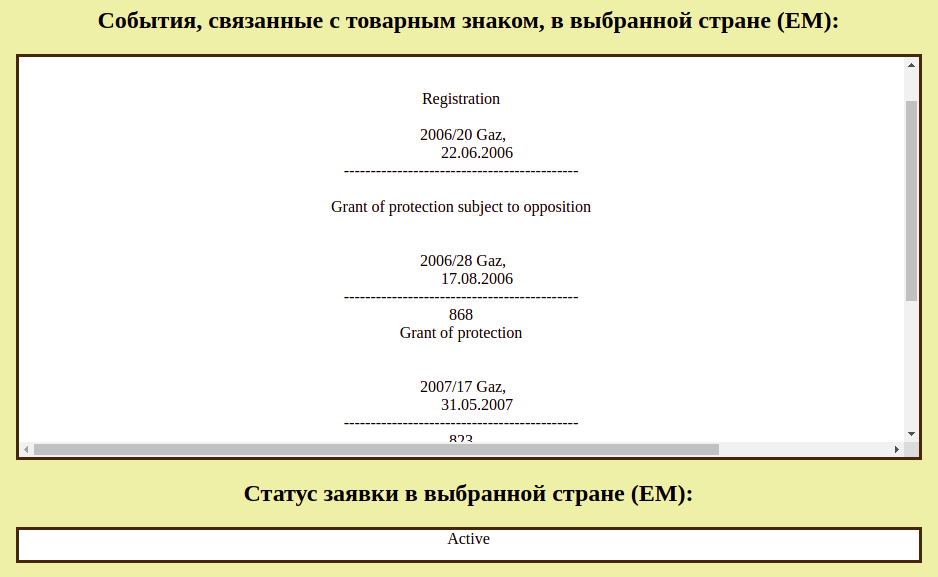
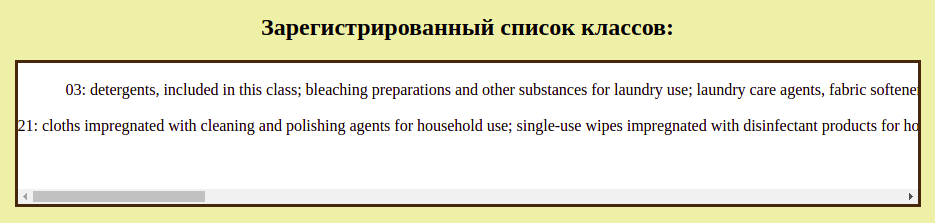
10. Хеник Б. HTML и CSS: путь к совершенству. Спб.: Питер, 2011.

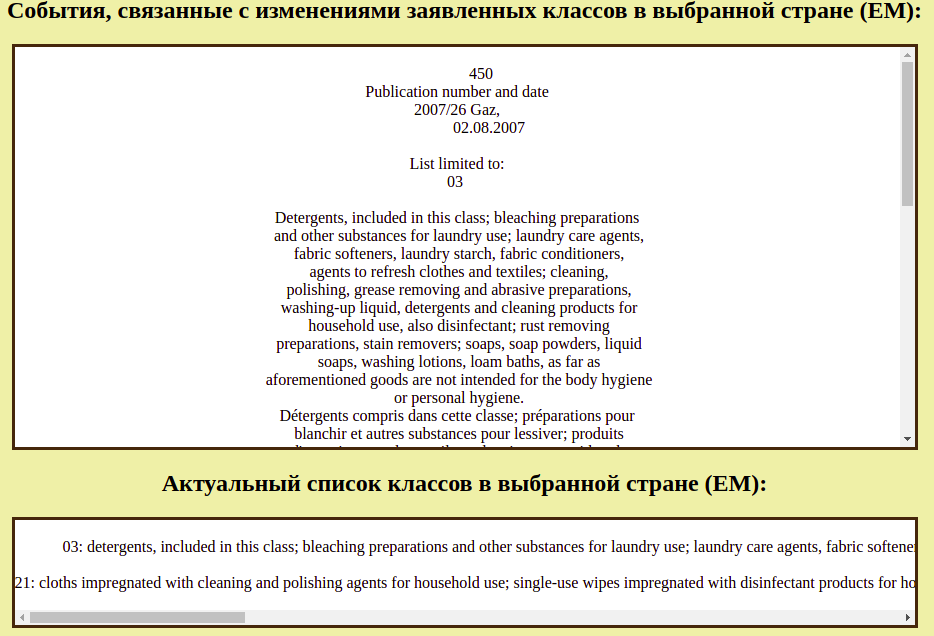
# Приложение 1

Внешний вид клиентской части веб-сервиса









# Приложение 2

Содержимое файлов с исходным кодом

**Файл index.js:**

const config = require('./config.js');

const osmosis = require('osmosis');

const parser = require('node-html-parser');

const bodyParser = require('body-parser');

const express = require('express');

var Pool = require('phantomjs-pool').Pool;

var app = express();

var urlencodedParser = bodyParser.urlencoded({ extended: false });

app.set('view engine', 'ejs');

app.use('/public', express.static('public'));

var allActions = config.AllActions();

app.get('/', function(req, res) {

res.render('index');

});

app.get('/designations', urlencodedParser, function(req, res) {

if (req.query.number.match(/^[0-9]+([A-Z]|[0-9])$/g) == null) return res.send('\nНомер заявки введён некорректно.\n');

let url = 'https://www3.wipo.int/madrid/monitor/en/showData.jsp?ID=' + req.query.number;

let designCountries = [];

// Проверка существования заявки с введённым номером

let appListCountries = [];

let check = osmosis.get(url).find('#documentContent').set(['.markname']);

check.data(data => {

if (!data[0])

return res.send('\nВведён несуществующий номер заявки.\n');

})

// Поиск стран, в которые были поданы заявки

var appListCountriesPr10 = new Promise(resolve => {

osmosis

.get(url)

.find('#container-4')

.set(['.fragment-content .transaction .ligneBox .txt .text a'])

.data(function(data) {

appListCountries = data;

resolve(appListCountries);

})

});

// Создание выборки из обозначений стран

var designCountriesPr11 = new Promise(resolve => {

appListCountriesPr10.then(appListCountries => {

if (appListCountries != null)

{

for (let i = 0; i < appListCountries.length; i++)

{

let tmpStr = appListCountries[i];

let len = tmpStr.length;

let tmpDes = tmpStr[len - 3] + tmpStr[len - 2];

designCountries.push(tmpDes);

}

designCountries = designCountries.filter(function(item, pos) {

return designCountries.indexOf(item) == pos;

})

}

else

designCountries = null;

resolve(designCountries);

})

});

designCountriesPr11.then(designCountries => {

if (designCountries != null)

{

let ind = designCountries.indexOf('RU');

if (ind != -1)

designCountries.splice(0, 0, designCountries.splice(ind, 1)[0]);

res.render('designations', {desigs: designCountries, number: req.query.number});

console.log(`\nFound designations for ${req.query.number}.\n`);

}

else

return res.send('\nВведён несуществующий номер заявки.\n');

});

});

app.get('/search', urlencodedParser, function(req, res) {

if (req.query.number.match(/^[0-9]+([A-Z]|[0-9])$/g) == null) return res.send('\nНомер заявки введён некорректно.\n');

let chosenCountry = req.query.chosen\_desig;

let userGeneralMsg = 'Запрос к системе Madrid Monitor выполнен успешно!';

let url = 'https://www3.wipo.int/madrid/monitor/en/showData.jsp?ID=' + req.query.number;

function jobCallback(job, worker, index) {

if (index < 1)

{

job(

{

url : url,

chosenCountry : chosenCountry

},

function(err, data)

{

if (err)

{

userGeneralMsg = '\nОшибка запроса к сервису Madrid Monitor. Попробуйте повторить команду через несколько секунд.\n';

console.log(userGeneralMsg);

return res.send(userGeneralMsg);

}

else

{

let html = data.html;

let page = parser.parse(html);

//Регистрационный номер и наименование товарного знака

let nameOfTrademark;

nameOfTrademark = page.querySelector('#documentContent .markname').text;

//Дата регистрации

let regIndex,

dateOfRegistr,

datesOfRegistr = [],

datesOfRegistrCodes = [];

page.querySelectorAll('.box\_content:first-child .description .p .inidCode').forEach(function(data){

datesOfRegistrCodes.push(data.text);

});

regIndex = datesOfRegistrCodes.indexOf('151');

page.querySelectorAll('.box\_content:first-child .description .p .text').forEach(function(data){

datesOfRegistr.push(data.text);

});

if (datesOfRegistr[regIndex])

dateOfRegistr = datesOfRegistr[regIndex];

else

dateOfRegistr = 'В Madrid monitor информация отсутствует';

//Ожидаемая дата обновления регистрации знака

let renIndex,

dateOfRenewal,

datesOfRenewal = [],

datesOfRenewalCodes = [];

page.querySelectorAll('.box\_content:first-child .description .p .inidCode').forEach(function(data){

datesOfRenewalCodes.push(data.text);

});

renIndex = datesOfRenewalCodes.indexOf('180');

page.querySelectorAll('.box\_content:first-child .description .p .text').forEach(function(data){

datesOfRenewal.push(data.text);

});

if (datesOfRenewal[renIndex])

dateOfRenewal = datesOfRenewal[renIndex];

else

dateOfRenewal = 'В Madrid monitor информация отсутствует';

//Язык регистрации товарного знака

let countryRegIndex,

countryReg,

countriesReg = [],

countriesRegCodes = [];

page.querySelectorAll('.box\_content:first-child .description .p .inidCode').forEach(function(data){

countriesRegCodes.push(data.text);

});

countryRegIndex = countriesRegCodes.indexOf('270');

page.querySelectorAll('.box\_content:first-child .description .p .text').forEach(function(data){

countriesReg.push(data.text);

});

if (countriesReg[countryRegIndex])

countryReg = countriesReg[countryRegIndex];

else

countryReg = 'В Madrid monitor информация отсутствует';

//Сопоставление кодов и текстов второго блока

// Получение кодов и их индексов во втором блоке

let codesIndexesArr = [[], []];

page.querySelectorAll(`#fragment-detail > div.fragment-content.box\_content.retreci > div > div .inidCode`).forEach(function(data, i) {

codesIndexesArr[0].push(i + 1);

codesIndexesArr[1].push(data.text);

});

// Получение текстов и их индексов во втором блоке

let textsIndexesArr = [[], []];

page.querySelectorAll(`#fragment-detail > div.fragment-content.box\_content.retreci > div > div .text`).forEach(function(data, i) {

textsIndexesArr[0].push(i + 1);

textsIndexesArr[1].push(data.text);

});

// Соединение массивов по совпадающим индексов в один (итоговое сопоставление)

let codesTextsArr = [[], []];

let codesIndexesArr\_len = codesIndexesArr[0].length;

for (let i = 0; i < codesIndexesArr\_len; ++i)

{

for (let j = 0; j < codesIndexesArr\_len; ++j)

{

if (codesIndexesArr[0][i] == textsIndexesArr[0][j])

{

codesTextsArr[0].push(codesIndexesArr[1][i]);

codesTextsArr[1].push(textsIndexesArr[1][j]);

break ;

}

}

}

//Детали владельца товарного знака

let holderDetails;

let holderIndex = codesTextsArr[0].indexOf('732');

if (codesTextsArr[1][holderIndex])

holderDetails = codesTextsArr[1][holderIndex];

else

holderDetails = 'В Madrid monitor информация отсутствует';

//Детали представителя товарного знака

let representativeDetails;

let representativeIndex = codesTextsArr[0].indexOf('740');

if (codesTextsArr[1][representativeIndex])

representativeDetails = codesTextsArr[1][representativeIndex];

else

representativeDetails = 'В Madrid monitor информация отсутствует';

//Зарегистрированный список классов

let Classes = []

let tmpClassesText = [];

page.querySelectorAll('.retreci .description .p .text > dl > dd .firstLanguage').forEach(function(el) {

tmpClassesText.push(el.text);

});

let tmpClassesNumbers = [];

page.querySelectorAll('.retreci .description .p .text > dl > dt').forEach(function(el) {

tmpClassesNumbers.push(el.text);

});

let regCl\_j = 0;

for (let i = 0; i < tmpClassesNumbers.length; i++)

{

Classes[regCl\_j] = [];

Classes[regCl\_j].push(tmpClassesNumbers[i].replace(/\s+/g, ' ').trim());

Classes[regCl\_j].push(tmpClassesText[i].toLowerCase().split('.')[0].replace(/\s+/g, ' ').trim());

++regCl\_j;

}

let regClassesOutList = '';

for (let i = 0; i < Classes.length; i++)

regClassesOutList += Classes[i][0] + ': ' + Classes[i][1] + '\n\n';

//События, связанные с товарным знаком, в выбранной стране

// Поиск стран, в которые были поданы заявки

let appListCountries = [];

page.querySelectorAll('#container-4 .fragment-content .transaction .ligneBox .txt .text a').forEach(function(data) {

appListCountries.push(data.text);

});

// Создание выборки из обозначений стран

let designCountries = [];

for (let i = 0; i < appListCountries.length; i++)

{

let tmpStr = appListCountries[i].replace(/\s+/g, ' ').trim();

let len = tmpStr.length;

let tmpDes = tmpStr[len - 3] + tmpStr[len - 2];

designCountries.push(tmpDes);

}

designCountries = designCountries.filter(function(item, pos) {

return designCountries.indexOf(item) == pos;

});

// Выбор страны по совпадающему обозначению

let countryNumber = designCountries.indexOf(chosenCountry) + 1;

// События по выбранной стране

let tmpEventsForCountries = [];

let eventsForCountries;

page.querySelectorAll(`#container-4 .fragment-content .transaction:nth-child(${countryNumber + 1}) .description .box\_content .retreci`).forEach(function(data) {

tmpEventsForCountries.push(data.text.replace(/\n\n\n\n+/g, '\n--------------------------------------------\n'));

});

eventsForCountries = tmpEventsForCountries[0];

lenEvForCountr = eventsForCountries.split('\n--------------------------------------------\n').length;

//Определение статуса заявки

// Поиск статуса по последнему значимому событию

let eventsForStatusCountry = [];

page.querySelectorAll(`#container-4 .fragment-content .transaction:nth-child(${countryNumber + 1}) .description .box\_content .retreci .inidText`).forEach(function(data, i) {

if (eventsForCountries.indexOf(data.text) != -1 && i < lenEvForCountr)

eventsForStatusCountry.push(data.text);

});

let appStatusInCountry = 'Not found';

let flag = 0;

for (let i = eventsForStatusCountry.length - 1; i >= 0; i--)

{

for (let j = 0; j < allActions.length; j++)

{

if (eventsForStatusCountry[i] == allActions[j][0])

{

if (allActions[j][1] == 'i')

{

appStatusInCountry = 'Inactive';

flag = 1;

break ;

}

else if (allActions[j][1] == 'a')

{

appStatusInCountry = 'Active';

flag = 1;

break ;

}

else if (allActions[j][1] == 'r')

{

let tmpNoNotifyCheck = page.querySelectorAll(`#container-4 .fragment-content .transaction:nth-child(${countryNumber + 1}) .description .box\_content .retreci .text`)[1].text;

if (tmpNoNotifyCheck && tmpNoNotifyCheck != undefined && tmpNoNotifyCheck == 'The refusal period has expired and no notification of provisional refusal has been recorded (application of Rule 5 preserved)')

appStatusInCountry = 'Active';

else

appStatusInCountry = 'Inactive';

flag = 1;

break ;

}

else if (allActions[j][1] == 'n')

{

if (i == 0)

{

appStatusInCountry = 'Inactive';

flag = 1;

}

continue;

}

}

else if (eventsForStatusCountry[i].indexOf('18ter(3)') != -1)

{

appStatusInCountry = 'Inactive';

flag = 1;

break ;

}

else if (eventsForStatusCountry[i].indexOf('18ter(1)') != -1 || eventsForStatusCountry[i].indexOf('18ter(2)') != -1)

{

appStatusInCountry = 'Active';

flag = 1;

break ;

}

}

if (flag == 1)

break ;

}

if (appStatusInCountry == 'Active')

{

let strCurDate = new Date().toLocaleString("en-US", {timeZone: "Europe/Moscow"});

let mmddyyCur = strCurDate.split(',')[0].split('/');

for (let i = 0; i < mmddyyCur.length; ++i)

{

if (Number(mmddyyCur[i]) >= 1 && Number(mmddyyCur[i]) <= 9)

mmddyyCur[i] = '0' + mmddyyCur[i];

}

let curDate = new Date(mmddyyCur[2] + '-' + mmddyyCur[0] + '-' + mmddyyCur[1] + 'T00:00:00.000Z');

let ddmmyyRen = dateOfRenewal.split('.');

let expectedRenewalDate = new Date(ddmmyyRen[2] + '-' + ddmmyyRen[1] + '-' + ddmmyyRen[0] + 'T00:00:00.000Z');

if (curDate.getTime() > expectedRenewalDate.getTime())

{

eventsForCountries += '\n==========================================================\n';

eventsForCountries += 'Срок действия защиты товарного знака истёк.\nОбновление регистрации товарного знака не было осуществлено\nдо ожидаемой даты обновления регистрации товарного знака.';

appStatusInCountry = 'Inactive';

}

}

//Определение списка классов, одобренного в выбранной стране

// Определение изменённого списка классов

let outClassActions = 'Не найдено.';

let outClassList = 'Не найдено.';

let flagFoundActions = 0;

let detailEventsCountryClass = [];

let j = 0;

// Определение событий, связанных с изменением списка классов

for (let i = 4; i <= page.querySelectorAll('.transaction.fragment-content.onlyWhenActive .description.box\_content.retreci').length + 3; ++i)

{

let data = page.querySelector(`.transaction.fragment-content.onlyWhenActive.open:nth-child(${i}) .description.box\_content.retreci`);

if (data != null && (data.text.toLowerCase().indexOf('class') != -1

|| data.text.toLowerCase().indexOf('goods') != -1

|| data.text.toLowerCase().indexOf('services') != -1

|| data.text.toLowerCase().indexOf('for which protection of the mark is granted') != -1

|| data.text.toLowerCase().indexOf('specification') != -1

|| data.text.toLowerCase().match(/list(.|\n)\*limit/) != null

|| data.text.toLowerCase().match(/limit(.|\n)\*list/) != null

|| data.text.toLowerCase().match(/list(.|\n)\*delete/) != null

|| data.text.toLowerCase().match(/delete(.|\n)\*list/) != null))

{

flagFoundActions = 1;

detailEventsCountryClass[j] = [];

detailEventsCountryClass[j].push(data.text.replace(/\n+/g, '\n').trim());

detailEventsCountryClass[j].push(i);

++j;

}

}

if (flagFoundActions == 1)

outClassActions = detailEventsCountryClass.join('\n--------------------------------------------\n\n');

else

outClassActions = 'Не найдены события, связанные с изменениями заявленных классов.';

// Передача полученных событий в функцию определения изменённого списка классов

if (flagFoundActions == 1 && appStatusInCountry == 'Active')

{

let {msgClassFunc, arrNewClassRes} = actualClassListFunc(page, detailEventsCountryClass, Classes, flagFoundActions);

if (msgClassFunc == 'new classes')

{

let tmpClListArr = [];

for (let i = 0; i < arrNewClassRes.length; ++i)

{

let tmpClListStr = arrNewClassRes[i].join(': ');

tmpClListArr.push(tmpClListStr);

}

outClassList = tmpClListArr.join('\n\n');

}

else if (msgClassFunc == 'no classes left' || msgClassFunc == 'inactive app'

|| msgClassFunc == 'class actions' || msgClassFunc == 'unable to handle actions')

outClassList = arrNewClassRes;

}

else if (appStatusInCountry == 'Inactive')

outClassList = 'Регистрация неактивна в данной стране, защита не предоставляется ни одному классу.';

else

outClassList = 'Не найдены события, связанные с изменениями заявленных классов.\nАктуальный перечень классов соответствует заявленному при регистрации.\nСм. ЗАРЕГИСТРИРОВАННЫЙ СПИСОК КЛАССОВ.';

//Передача данных в html и вывод результата на сервисе

res.render('search', {

chosenCountry: chosenCountry,

userGeneralMsg: userGeneralMsg,

nameOfTrademark: nameOfTrademark,

dateOfRegistr: dateOfRegistr,

dateOfRenewal: dateOfRenewal,

countryReg: countryReg,

holderDetails: holderDetails.replace(/\n\n\n+/g, '').trim(),

representativeDetails: representativeDetails.replace(/\n\n\n+/g, '').trim(),

regClassesOutList: regClassesOutList,

eventsForCountries: eventsForCountries,

appStatusInCountry: appStatusInCountry,

outClassActions: outClassActions,

outClassList: outClassList,

url: url

});

designCountries = [];

Classes = [];

chosenCountry = '';

userGeneralMsg = 'Запрос к системе Madrid Monitor выполнен успешно!';

nameOfTrademark = '';

dateOfRegistr = '';

dateOfRenewal = '';

countryReg = '';

holderDetails = '';

representativeDetails = '';

regClassesOutList = '';

eventsForCountries = '';

outClassActions = '';

outClassList = '';

console.log(`\nWork completed successfully for ${req.query.number}!\n`);

}

});

}

else

job(null);

}

let pool = new Pool({

numWorkers : 1,

jobCallback : jobCallback,

workerFile : \_\_dirname + '/worker.js'

});

pool.start();

});

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => { console.log(`Server is listening on port ${PORT}.`); });

// Функция определения изменённого списка классов

function actualClassListFunc(page, detailEventsCountryClass, Classes, flagFoundActions)

{

let newClasses = Classes;

// Проход по каждому подробному событию, связанному с изменённым списком классов, для выбранной страны

for (let d\_ind = 0; d\_ind < detailEventsCountryClass.length; ++d\_ind)

{

if (detailEventsCountryClass[d\_ind][0].indexOf('Delete from list') != -1

|| detailEventsCountryClass[d\_ind][0].indexOf('List limited to') != -1

|| detailEventsCountryClass[d\_ind][0].indexOf('for which protection of the mark is granted') != -1)

{

let data1 = page.querySelectorAll(`.transaction.fragment-content.onlyWhenActive:nth-child(${detailEventsCountryClass[d\_ind][1]}) .description.box\_content.retreci .inid`);

for (let d1\_ind = 0; d1\_ind < data1.length; ++d1\_ind)

{

if (data1[d1\_ind].text.indexOf('Delete from list') != -1)

{

flagFoundActions = 2;

let changeClasses = [];

let iter = data1[d1\_ind].nextElementSibling.firstChild.firstChild;

let flag = 0;

while (iter != undefined)

{

let arrInd = Math.floor(flag / 2);

if (flag % 2 == 0)

{

changeClasses[arrInd] = [];

changeClasses[arrInd].push(iter.text.replace(/\s+/g, ' ').trim());

}

else

changeClasses[arrInd].push(iter.text.toLowerCase().split('.')[0].replace(/\s+/g, ' ').trim());

iter = iter.nextElementSibling;

++flag;

}

for (let ch\_ind = 0; ch\_ind < changeClasses.length; ++ch\_ind)

{

let listDel = changeClasses[ch\_ind][1].split(';');

for (let ind = 0; ind < newClasses.length; ++ind)

{

if (changeClasses[ch\_ind][0] == newClasses[ind][0])

{

let listCl = newClasses[ind][1].split(';');

for (let ind1 = 0; ind1 < listDel.length; ++ind1)

{

let actFlag = 0;

for (let i = 0; i < listCl.length; ++i)

{

if (listDel[ind1].replace(/\s+/g, ' ').trim() == listCl[i].replace(/\s+/g, ' ').trim())

{

listCl.splice(i, 1);

actFlag = 1;

break ;

}

}

if (actFlag == 0)

return {msgClassFunc: 'class actions', arrNewClassRes: 'Невозможно корректно определить новый список классов. Смотрите cобытия, связанные\n с изменениями заявленных классов.'};

}

newClasses[ind][1] = listCl.join(';').replace(/\s+/g, ' ').trim();

}

}

}

}

if (data1[d1\_ind].text.indexOf('List limited to') != -1)

{

flagFoundActions = 2;

let changeClasses = [];

let iter = data1[d1\_ind].nextElementSibling.firstChild.firstChild;

let flag = 0;

while (iter != undefined)

{

let arrInd = Math.floor(flag / 2);

if (flag % 2 == 0)

{

changeClasses[arrInd] = [];

changeClasses[arrInd].push(iter.text.replace(/\s+/g, ' ').trim());

}

else

changeClasses[arrInd].push(iter.text.toLowerCase().split('.')[0].replace(/\s+/g, ' ').trim());

iter = iter.nextElementSibling;

++flag;

}

for (let ch\_ind = 0; ch\_ind < changeClasses.length; ++ch\_ind)

{

for (let ind = 0; ind < newClasses.length; ++ind)

{

if (changeClasses[ch\_ind][0] == newClasses[ind][0])

newClasses[ind][1] = changeClasses[ch\_ind][1];

}

}

}

if (data1[d1\_ind].text.indexOf('for which protection of the mark is granted') != -1)

{

flagFoundActions = 2;

let changeClasses = [];

let iter = data1[d1\_ind].nextElementSibling.firstChild.firstChild;

let flag = 0;

while (iter != undefined)

{

let arrInd = Math.floor(flag / 2);

if (flag % 2 == 0)

{

changeClasses[arrInd] = [];

changeClasses[arrInd].push(iter.text.replace(/\s+/g, ' ').trim());

}

else

changeClasses[arrInd].push(iter.text.toLowerCase().split('.')[0].replace(/\s+/g, ' ').trim());

iter = iter.nextElementSibling;

++flag;

}

newClasses = changeClasses;

}

}

}

let strToParse = null;

strToParse = detailEventsCountryClass[d\_ind][0].replace(/\s+/g, ' ').trim().toLowerCase()

.match(/refus(.|\n)\*for all(.|\n)\*in class(.|\n)\*?[;|\.]/);

if (strToParse != null)

{

flagFoundActions = 2;

let arrNumClassesToDel = strToParse[0].match(/(\d+)/g);

for (let i = 0; i < arrNumClassesToDel.length; ++i)

{

for (let j = 0; j < newClasses.length; ++j)

{

if (arrNumClassesToDel[i] == Number(newClasses[j][0]))

newClasses.splice(j, 1);

}

}

}

strToParse = null;

strToParse = detailEventsCountryClass[d\_ind][0].replace(/\s+/g, ' ').trim().toLowerCase()

.match(/invalid(.|\n)\*for all(.|\n)\*in class(.|\n)\*?[;|\.]/);

if (strToParse != null)

{

flagFoundActions = 2;

let arrNumClassesToDel = strToParse[0].match(/(\d+)/g);

for (let i = 0; i < arrNumClassesToDel.length; ++i)

{

for (let j = 0; j < newClasses.length; ++j)

{

if (arrNumClassesToDel[i] == Number(newClasses[j][0]))

newClasses.splice(j, 1);

}

}

}

strToParse = null;

strToParse = detailEventsCountryClass[d\_ind][0].replace(/\s+/g, ' ').trim().toLowerCase()

.match(/class(.|\n)\*cancell/);

if (strToParse != null)

{

flagFoundActions = 2;

let arrNumClassesToDel = strToParse[0].match(/(\d+)/g);

for (let i = 0; i < arrNumClassesToDel.length; ++i)

{

for (let j = 0; j < newClasses.length; ++j)

{

if (arrNumClassesToDel[i] == Number(newClasses[j][0]))

newClasses.splice(j, 1);

}

}

}

}

if (flagFoundActions == 2)

{

if (newClasses.length != 0)

return {msgClassFunc: 'new classes', arrNewClassRes: newClasses};

else

return {msgClassFunc: 'no classes left', arrNewClassRes: 'Защита не предоставляется ни одному классу.'};

}

else

return {msgClassFunc: 'unable to handle actions', arrNewClassRes: 'Невозможно корректно определить новый список классов. Смотрите cобытия, связанные\n с изменениями заявленных классов.'};

}

**Файл config.js:**

const AllActions = () => {

return ([

['No request for review or appeal has been lodged', 'i'],

['A request for review or an appeal has been lodged', 'i'],

['Withdrawal of request for review or appeal', 'i'],

['Cancellation effected for some of the goods and services at the request of an Office of origin in accordance with Article 6(4) of the Agreement or Article 6(4) of the Protocol', 'n'],

['Cancellation effected for all the goods and services at the request of an Office of origin in accordance with Article 6(4) of the Agreement or Article 6(4) of the Protocol', 'i'],

['Judicial actions or proceedings under Rule 22(1)(b)', 'n'],

['Continuation of effect', 'r'],

['Cancellation of the Recording of a License', 'n'],

['Partial assignment', 'a'],

['Partial assignment to specific Contracting Parties', 'a'],

['Partial assignment of particular Goods and Services', 'a'],

['Division/Merger of basic registration', 'n'],

['Accepted with reserve', 'a'],

['Designated contracting party(ies) for which the second installment has been paid (Rule 40(3))', 'n'], //тонкий момент для определения объема охраны

['Designated contracting party(ies) for which the second installment has not been paid (Rule 40(3))', 'i'],

['Registration', 'r'],

['Subsequent designation', 'r'],

['Replacement of national registration by an international registration', 'n'],

['Final decision confirming the refusal of protection', 'i'],

['Disclaimer', 'i'],

['Final decision confirming the disclaimer', 'i'],

['Statement indicating that protection of the mark is protected for some of the goods and services requested', 'a'],

['Statement indicating that protection of the mark is refused for all the goods and services requested', 'i'],

['Statement indicating that the mark is protected for all the goods and services requested', 'a'],

['Further statement under Rule 18ter(4) indicating the goods and services for which the mark is protected', 'a'],

['Further statement under Rule 18ter(4) indicating that protection of the mark is refused for all the goods and services requested', 'i'],

['Further statement under Rule 18ter(4) indicating that protection of the mark is granted for all the goods and services requested', 'a'],

['Merger of international registrations', 'n'],

['Statement of grant of protection made under Rule 18ter(1) where no notification of provisional refusal has been communicated', 'a'],

['Grant of protection', 'a'],

['Grant of protection subject to opposition', 'i'],

['Restriction of the holder\'s right of disposal of the international registration', 'n'],

['Partial invalidation', 'a'],

['Total invalidation', 'i'],

['Ex Officio examination completed but opposition or observations by third parties still possible, under Rule 18bis(1)', 'i'],

['Limitation', 'n'],

['Change in the name or address of the licensee', 'n'],

['Declaration that a limitation has no effect', 'n'],

['New licence', 'n'],

['Subsequent designations resulting from Conversion (Rule 24(7))', 'r'],

['Opposition possible after the 18 months time limit', 'i'],

['Cancellation effected with respect to a designated Contracting Party following the non payment of the second part of the individual fee (Rule 34(3))', 'i'],

['Cancellation effected for some of the goods and services at the request of the holder under Rule 25', 'n'],

['Statement of grant of protection following a provisional refusal under Rule 18ter(2)(ii)', 'a'],

['Statement of grant of protection following a provisional refusal (acceptation with reserve) under Rule 18ter(2)(ii)', 'a'],

['Confirmation of total provisional refusal under Rule 18ter(3)', 'i'],

['Statement of grant of protection following a provisional refusal under Rule 18ter(2)(i)', 'a'],

['Cancellation effected for all the goods and services at the request of the holder under Rule 25', 'i'],

['Complementary renewal', 'a'],

['Renewal', 'a'],

['Designated contracting party(ies) which has not been the subject of a renewal (Rule 31(4)(a))', 'i'],

['Designated contracting party(ies) which has not been the subject of a renewal (Rule 31(4)(b))', 'i'],

['Partial refusal of protection', 'i'],

['Total refusal of protection', 'i'],

['Removal of restriction of the holder\'s right of disposal of the international registration', 'n'],

['Renunciation', 'i'],

['Declaration that a change in ownership has no effect', 'n'],

['Balances of fees', 'n'],

['Seniority Claimed Subsequent to the International Registration', 'n'],

['Final refusal of seniority claim', 'n'],

['Other final decision - Awaiting translation before publication', 'a'],

['Partial provisional refusal of protection - Awaiting translation before publication', 'i'],

['Partial provisional refusal of protection', 'i'],

['Statement indicating the goods and services for which protection of the mark is granted under Rule 18ter(2)(ii)', 'a']

]);

}

exports.AllActions = AllActions;

**Файл worker.js:**

var page = require('webpage').create();

module.exports = function(data, done, worker) {

page.open(data.url, function() {

var htmlForParse = page.evaluate(function(selector){

document.getElementById(selector).click();

return document.documentElement.innerHTML;

}, 'button' + data.chosenCountry);

done(null, {html: htmlForParse});

});

};

**Файл package.json:**

{

"name": "madridmonitor",

"version": "1.0.0",

"description": "",

"engines": {

"node": "10.19.0"

},

"main": "index.js",

"scripts": {

"start": "node index"

},

"author": "",

"license": "ISC",

"dependencies": {

"body-parser": "^1.19.0",

"ejs": "^3.1.6",

"express": "^4.16.4",

"node-html-parser": "^3.1.5",

"osmosis": "^1.1.10",

"phantomjs-pool": "^0.3.1",

"phantomjs2": "^2.2.0"

}

}

**Файл style.css, лежащий в папке public:**

#body {

background-color: rgb(243,229,171);

text-align: center;

}

#main\_page, #madrid\_link {

display: inline-block;

text-decoration: none;

text-transform: uppercase;

letter-spacing: 1px;

margin: 15px 25px;

padding: 15px 20px;

font-size: 20px;

font-weight: bold;

font-family: 'Montserrat', sans-serif;

transition: 0.4s ease-in-out;

border-radius: 10px;

color: #40382D;

box-shadow: inset 0 -5px 0 #40382D;

}

#main\_page:hover {

box-shadow: inset 0 -60px 0 #40382D, 2px 2px 3px rgba(0, 0, 0, 0.3);

color: #DECDA5;

}

#madrid\_link:hover {

box-shadow: inset 0 -60px 0 #40382D, 2px 2px 3px rgba(0, 0, 0, 0.3);

color: #DECDA5;

}

#service\_title {

text-align: center;

}

#application {

text-align: center;

}

#chosen\_country {

text-align: center;

}

#application\_confirmed {

text-align: center;

font-weight: bolder;

}

#application\_confirmed\_h {

text-align: center;

font-weight: bolder;

}

#userGeneralMsg {

height: 30px;

width: 900px;

/\*white-space: pre;\*/

margin-left: auto;

margin-right: auto;

text-align: center;

background: #fff;

border: 3px solid #452612;

overflow: auto;

}

#nameOfTrademark {

height: 30px;

width: 900px;

/\*white-space: pre;\*/

margin-left: auto;

margin-right: auto;

text-align: center;

background: #fff;

border: 3px solid #452612;

overflow: auto;

}

#dateOfRegistr {

height: 30px;

width: 900px;

/\*white-space: pre;\*/

margin-left: auto;

margin-right: auto;

text-align: center;

background: #fff;

border: 3px solid #452612;

overflow: auto;

}

#dateOfRenewal {

height: 30px;

width: 900px;

/\*white-space: pre;\*/

margin-left: auto;

margin-right: auto;

text-align: center;

background: #fff;

border: 3px solid #452612;

overflow: auto;

}

#countryReg {

height: 30px;

width: 900px;

/\*white-space: pre;\*/

margin-left: auto;

margin-right: auto;

text-align: center;

background: #fff;

border: 3px solid #452612;

overflow: auto;

}

#holderDetails {

/\*height: 90px;\*/

width: 900px;

white-space: pre;

margin-left: auto;

margin-right: auto;

text-align: center;

background: #fff;

border: 3px solid #452612;

overflow: auto;

}

#representativeDetails {

/\*height: 90px;\*/

width: 900px;

white-space: pre;

margin-left: auto;

margin-right: auto;

text-align: center;

background: #fff;

border: 3px solid #452612;

overflow: auto;

}

#regClassesOutList {

/\*height: 400px;\*/

max-height: 400px;

width: 900px;

white-space: pre;

margin-left: auto;

margin-right: auto;

text-align: center;

background: #fff;

border: 3px solid #452612;

overflow: auto;

}

#eventsForCountries {

/\*height: 270px;\*/

max-height: 400px;

width: 900px;

white-space: pre;

margin-left: auto;

margin-right: auto;

text-align: center;

background: #fff;

border: 3px solid #452612;

overflow: auto;

}

#appStatusInCountry {

height: 30px;

width: 900px;

/\*white-space: pre;\*/

margin-left: auto;

margin-right: auto;

text-align: center;

background: #fff;

border: 3px solid #452612;

overflow: auto;

}

#outClassActions {

max-height: 400px;

width: 900px;

white-space: pre;

margin-left: auto;

margin-right: auto;

text-align: center;

background: #fff;

border: 3px solid #452612;

overflow: auto;

}

#outClassList {

/\*height: 400px;\*/

max-height: 400px;

width: 900px;

white-space: pre;

margin-left: auto;

margin-right: auto;

text-align: center;

background: #fff;

border: 3px solid #452612;

overflow: auto;

}

**Файл index.ejs, лежащий в папке views:**

<!DOCTYPE html>

<html>

<head>

<meta http-equiv=Content-Type content="text/html;charset=utf-8">

<title>Madrid Monitor search</title>

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

</head>

<body id="body">

<div id = "service\_title">

<h1 id="service\_title\_h1">Поиск информации о товарном знаке в системе Madrid Monitor</h1>

</div>

<div id="application">

<form method="get" action="/designations" id = "application\_form">

<label id="app\_num"><strong>Введите номер заявки:</strong> <br>

<input id = 'input' type="text" name='number'>

</label>

<input id = 'button1' type="submit" value="Подтвердить">

</form>

</div>

</body>

</html>

**Файл designations.ejs, лежащий в папке views:**

<!DOCTYPE html>

<html>

<head>

<meta http-equiv=Content-Type content="text/html;charset=utf-8">

<title>Madrid Monitor search</title>

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

</head>

<body id="body">

<div id = "service\_title">

<h1 id="service\_title\_h1">Поиск информации о товарном знаке в системе Madrid Monitor</h1>

</div>

<div id="chosen\_country">

<form method="get" action="/search" id="designation\_form">

<p id="application\_confirmed\_h">Номер заявки:</p>

<select id="application\_confirmed" name="number">

<option><%= number %></option>

</select>

<p>Из доступных обозначений выберите обозначение той страны, в которой хотите узнать статус защиты товарного знака</p>

<select id="chosen\_country\_select" name="chosen\_desig">

<% desigs.forEach(function(item) { %>

<option><%= item %></option>

<% }); %>

</select>

<input id = 'button2' type="submit" value="Поиск">

</form>

</div>

</body>

</html>

**Файл search.ejs, лежащий в папке views:**

<!DOCTYPE html>

<html>

<head>

<meta http-equiv=Content-Type content="text/html;charset=utf-8">

<title>Madrid Monitor search</title>

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

<script>

function setMadridLink() {

let params = (new URL(document.location)).searchParams;

let appNumber = params.get("number");

let link = document.getElementById("madrid\_a\_href");

link.target = "\_blank";

link.href = "https://www3.wipo.int/madrid/monitor/en/showData.jsp?ID=" + appNumber;

link.click();

}

</script>

</head>

<body id="body">

<div id="div\_links">

<ul id="ul\_links">

<li id="main\_page">

<a id="main\_a\_href" href="/">Главная страница</a>

</li>

<li id="madrid\_link">

<a id="madrid\_a\_href" href="javascript:setMadridLink()">Страница товарного знака в Madrid Monitor</a>

</li>

</ul>

</div>

<div id="service\_title">

<h1 id="service\_title\_h1">Поиск информации о товарном знаке в системе Madrid Monitor</h1>

</div>

<h2>Сообщение пользователю:</h2>

<div id="userGeneralMsg">

<%= userGeneralMsg %>

</div>

<h2>Регистрационный номер и наименование товарного знака:</h2>

<div id="nameOfTrademark">

<%= nameOfTrademark %>

</div>

<h2>Дата регистрации товарного знака:</h2>

<div id="dateOfRegistr">

<%= dateOfRegistr %>

</div>

<h2>Ожидаемая дата обновления регистрации товарного знака:</h2>

<div id="dateOfRenewal">

<%= dateOfRenewal %>

</div>

<h2>Язык регистрации товарного знака:</h2>

<div id="countryReg">

<%= countryReg %>

</div>

<h2>Детали владельца товарного знака:</h2>

<div id="holderDetails">

<%= holderDetails %>

</div>

<h2>Детали представителя товарного знака:</h2>

<div id="representativeDetails">

<%= representativeDetails %>

</div>

<h2>Зарегистрированный список классов:</h2>

<div id="regClassesOutList">

<%= regClassesOutList %>

</div>

<h2>События, связанные с товарным знаком, в выбранной стране (<%=chosenCountry%>):</h2>

<div id="eventsForCountries">

<%= eventsForCountries %>

</div>

<h2>Статус заявки в выбранной стране (<%=chosenCountry%>):</h2>

<div id="appStatusInCountry">

<%= appStatusInCountry %>

</div>

<h2>События, связанные с изменениями заявленных классов в выбранной стране (<%=chosenCountry%>):</h2>

<div id="outClassActions">

<%= outClassActions %>

</div>

<h2>Актуальный список классов в выбранной стране (<%=chosenCountry%>):</h2>

<div id="outClassList">

<%= outClassList %>

</div>

</body>

</html>