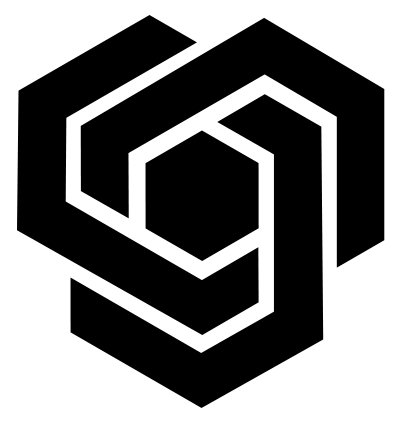
**Технически университет – София**

Факултет по Компютърни системи и технологии



Дипломна работа

Приложение за търсене на зависимост на поведение на потребител

Изготвил:

Елисавета Петрова Тодорова - фак.№ 121215002

Научен ръководител:

Доц. д-р инж. Аделина Алексиева-Петрова

Съдържание:

1. Описание на проблемната област. …………………………………………… 3
2. Проектиране и архитектура на приложението. ……………………………... 6
3. Особености на програмната реализация. ………………………………….. 13
4. Ръководство за потребителя …………………………………………………. 27
   1. Приложение …………………………………………………………………. 37
   2. Използвани ресурси ……………………………………………………….. 50

1. Описание на проблемната област

Целта на системата е да предостави лесен и удобен начин за анализирането и визуализиране на действията на потребителите на система за електронно обучение „Moodle“.

Работата на системата се състои в анализ и обработка на данни, чието съдържание се съхранява в „log“ файлове, които са с много големи размери.

Процесът за обработката на данни с огромен размер е свързана с използването на множество компютърни ресурси и отнема много време, което в днешните времена е недопустимо.

„Big data problem“ се изявява в огромно количество данни генерирани от различни източници. Сетовете от данни не могат да бъдат събрани, съхранени или обработени използвайки стандартни методи, заради тяхното количество и сложност. Най-често данните биват генерирано от социалните мрежи, уеб услугите, бизнес приложенията, „log“ данни генерирани от машини и много други.

Затова съществуват множество устройства за обработка на тези данни като един от тях е Apache Hadoop.С помощта на тези устройства тези данни от най-различни източни в различен формат могат да бъда анализиран и обработени.

Анализът на тези данни е важен защото веднъж анализирани, например в медицината, те могат да помогна за предотвратяването да конкретни заболявания, като ги хванат в начален стадий на развитие. Също така е много полезно в банковия сектор където помага за разпознаването на незаконни действия като пране на пари. И последно, но не на последно място в метрологията за изучаването на глобалното затопляне.

[1] Според проучване на корпорацията EMC до 2020 количеството данни съхранявани в компютри, мобилни телефони в облаци, преносими памети и други, в световен мащаб се очаква да достигне 40 трилиона гигабайта данни.

[2] Според проучване на компанията IBM 90 процента от тези данни са били създадени през последните две години. През 2012 в интернет са съществували 2.5 милиарда интернет потребители. През 2014 броят на потребителите е достигнал 3 милиарда, а в днешно време през 2019 съществуват 4.1 милиарда интернет потребителя. Това потвърждава твърдението че количеството на данни в интернет се увеличава с експоненциална скорост.

[3] На един човек би му отнело 181 милиона година за да изтегли всичките данни намиращи се в интернет.

[4] Интернет потребителите генерират 2.5 квинтилиона данни всеки ден.

[5] От голямото количество от данни които се генерират всяка секунда няма никаква ползва ако те не бъдат маркирани или анализирани. Според проучване на IDC от 2012 година само 0,5 процента от данните са анализирани, като процентът на маркираните данни е малко по-висок: 3 процента. Други проучвания са стигнали до извода, че не всички данни имат потенциала да допринесат с някаква стойност.

През 2017 списание Economist твърди, данните ще заменят петролът като най-ценно нещо на света. Сравнението между данните и петрола води до заключението че трябва да се събира и съхранява колкото е възможно повече данни.

Според друга статистика от IDC през 2012 година само 22 процента от данните имат потенциала да бъдат анализирани. Това включва данни от различни сфери като видео наблюдение, шоубизнес, социални медии и други. Същият източник посочва, че до 2020 година процентът на данни, които са потенциален кандидат за анализ, ще скочи на 37.

[6] С нарастването на потребителите онлайн нараства и потребителите на социалните мрежи. За 2012 година потребителите са били около 3 милиона, а през 2016 година са станали 7 милиона. Времето прекарано онлайн в тези сайтове също нараства. През 2012 е било час и половина, а през 2016 – 2 часа и 15 минути. Освен в социалните мрежи, 16 процента от времето потребителите онлайн прекарват в гледане на телевизия, 16 % – слушане на музика, 13 % - четене на пресата, останалите 22 процента за резервирани за други дейности.

Вече все повече компании започват да инвестират в анализът на данни.

[7] Благодарение на анализа на данни Netflix, компания, доставчик на филми и сериали, успява да спести 1 милиард на година за задържане на своите клиенти, като подобрява алгоритъма си за това какво да им предлага с помощта на анализирани данни.

Тъй като проблемът с анализът на данни с големи размери е тема която обхваща много области, системата е направена по такъв начин че тя може много лесно да се разшири и промени за да може да бъде използвана в зависимост от типовете и количеството на данни които трябва да бъдат анализирани.

Приложението се състои от три компонента – това са „backend“ подсистема, която имплементира анализира данните, „ REST API“ което обработва вече анализираните данни във формат подходящ за да могат да бъдат визуализирани от уеб интерфейса. Той действа като връзка между  backend  и уеб интерфейсът. Уеб интерфейсът е компонентът, който администратора или потребителят използва за да види вече анализираните данни или да направи заявка за данни които не са все още анализирани. Предоставя се и възможността за визуализиране на конкретни данни за определена дата зададена от потребителя, като по този начин можем да разберем кои да най-активните времена в които се използва системата която подлагаме да анализ за да се предотврати евентуално претоварване на системата.

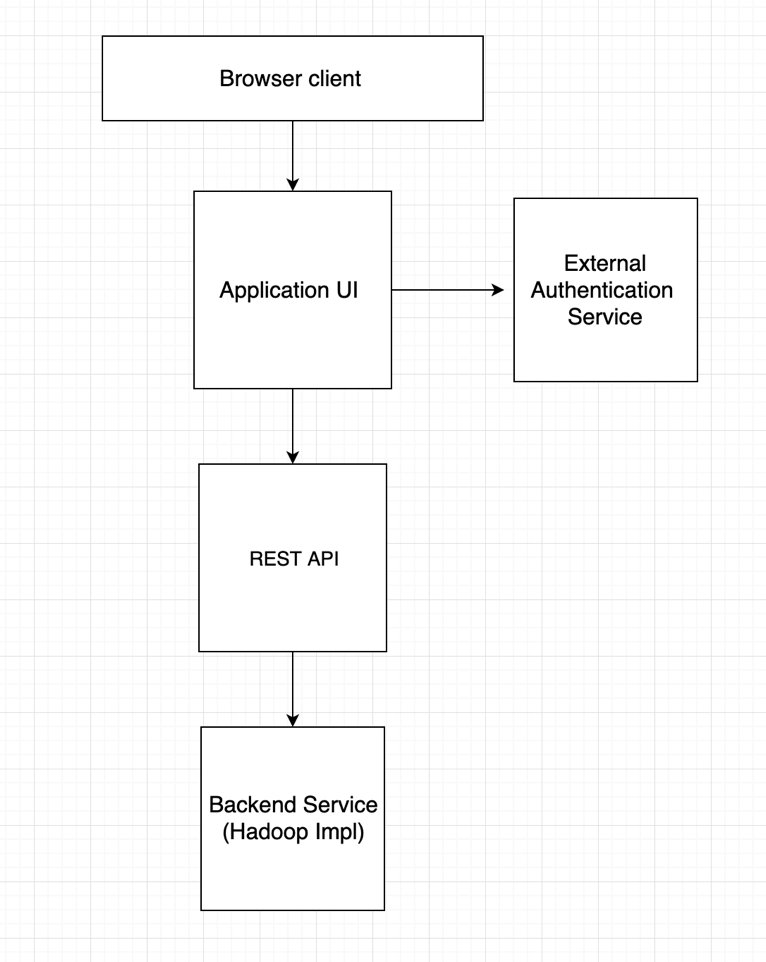
Приложението използва външен доставчик за автентикация, който използва протоколът SAML2.0. Предоставя се възможността за запазване текущата сесия на аврентицирания потребител и при затваряне на браузъра и повторно достъпване на системата не се налага потребителят да въвежда своята парола и потребителско име. При желание потребителят може да използва опцията за изход от приложението, като така той ще зачисти текущата сесия и при достъпване на приложението ще му бъдат изискани парола и потребителско име.

За реализирана на системата за анализиране на действието на потребител са използвани технологиите: JAVA, Apache Hadoop, Spring MVC, Spring boot, Bootstrap, Html, CSS, JavaScript, Thymeleaf.

За автентикация е използван продуктът на „SAP Identity and Authentication Service“.

2. Проектиране и архитектура на приложението.

Блок диаграма на архитектурата на приложението



Приложението се състои от три компонентна: „backend“ сървър, „Rest API“, уеб интерфейс, външен продукт за автентикация.

„Backend“ сървъра е ядрото на приложението. Той изпълнява работата по обработка на данни. Използва Apache Hadoop в „standalone mode“ за да стартира Мap Reduce джобове които обработват данни по предварително зададен начин. Данните за обработка, които използва приложението са логове от системата за електронно обучение Moodle.

Apache Hadoop е проект с отворен код на Apache фондацията, който улеснява използването на мрежа от много компютри за решаване на проблеми включващи огромни количества данни и изчисления. Той предоставя софтуерна рамка за разпределено съхранение и обработка на големи данни, използвайки модела на програмиране MapReduce. Първоначално проектиран за компютърни клъстери изградени от стоков хардуер, за което и до днес се използва, той също намира приложение и в клъстери от по-висок клас хардуер.

Всички модули на Hadoop са проектирани с предположението, че хардуерните повреди са често срещано явление, и това трябва да се обработи автоматично от Apache Hadoop.

Ядрото на Apache Hadoop се състои от част за съхранение известна като Hadoop разпределена файлова система (HDFS), и обработваща част, която в моделът за програмиране  MapReduce. Hadoop разпределя файловете на големи блокове и ги разделя по възли в клъстер. След това прехвърля пакетирания код във възли, за да обработва данните паралелно. Този подход се възползва от местоположението на данните, където възлите манипулират данни, до които имат достъп. Това позволява наборът от данни да се обработва по-бързо и по-ефективно, отколкото би бил в по- конвенционалната суперкомпютърна архитектура, която разчита на паралелна файлова система, където изчисленията и данните се разпределят чрез високоскоростна мрежа.

Продуктът на Apache Hadoop се състои от следните модули:

Hadoop Common – съдържа библиотеки и помощни програми, необходими от други модули на Hadoop.

Hadoop разпределена файлова система (HDFS) – разпределена файлова система, която съхранява данни на стокови машини, осигуряваща много висока обща честотна лента в целия клъстер. Тя е сърцето на Apache Hadoop.

Разпределената файлова система на Hadoop (HDFS) е основната система за съхранение на данни, използвана от Hadoop приложенията. Той използва NameNode и DataNode архитектура за имплементиране на разпределена файлова система, която осигурява висококачествен достъп до данни в силно мащабируеми клъстери на Hadoop.

HDFS е ключова част от много екосистемни технологии на Hadoop, тъй като предоставя надеждни средства за управление на пулове от големи данни и поддръжка на свързани с анализ на голяма количество данни приложения.

Начин на работа на HDFS.

HDFS  поддържа бързия трансфер на данни между изчислителните възли. В началото си тя е била тясно свързана с MapReduce, програмен модел за обработка на данни.

Когато HDFS приема данни, разбива информацията на отделни блокове и ги разпределя в различните възли в клъстер, като по този начин дава възможност за високоефективна паралелна обработка.

Нещо повече, разпределената файлова система Hadoop е специално проектирана да бъде устойчива на повреди. Файловата система копира или копира всеки фрагмент от данни няколко пъти и разпределя копията в отделни възли, поставяйки поне едно копие на различен сървърен багажник от останалите. В резултат на това данните за възлите, които се сриват, могат да бъдат намерени другаде в клъстера. Това гарантира, че обработката може да продължи, докато данните се възстановяват.

HDFS използва главна / подчинена архитектура. В първоначалното си въплъщение всеки клъстер Hadoop се състоеше от един NameNode, който управлява операциите на файловата система и поддържа DataNodes, които управляват съхранението на данни в отделни изчислителни възли. HDFS елементите се комбинират, за да поддържат приложения с големи масиви от данни.

Тази архитектура на "възел на данни" от основния възел приема за свой дизайн елементи от файловата система на Google (GFS), собствена файлова система, очертана в техническите документи на Google, както и от Общата паралелна файлова система (GPFS) на IBM, формат, който повишава I / O чрез премахване на блокове от данни на множество дискове, записване на паралелни блокове. Въпреки че HDFS не е съвместим с модела на интерфейса на операционната система, той в някои аспекти озвучава стила на POSIX.

Разпространената файлова система Hadoop възникна в Yahoo като част от изискванията за рекламиране и търсене на реклами на тази компания. Подобно на други уеб-ориентирани компании, Yahoo намери жонглиране на различни приложения, достъпни от все по-голям брой потребители, които създаваха все повече и повече данни. Facebook, eBay, LinkedIn и Twitter са сред уеб компаниите, които използват HDFS, за да подкрепят анализа на големи данни, за да отговорят на същите тези изисквания.

Но файловата система намери употреба извън това. HDFS се използва от The New York Times като част от мащабни преобразувания на изображения, Media6Degrees за обработка на журнали и машинно обучение, LiveBet за съхранение на журнали и анализ на коефициенти, Joost за анализ на сесии и Fox Audience Network за анализ на журнали и извличане на данни. HDFS също е в основата на много алтернативи за съхранение на данни с отворен код, понякога наричани езера с данни.

Тъй като HDFS обикновено се внедрява като част от много мащабни реализации, поддръжката на хардуер с ниска цена е особено полезна функция. Такива системи, работещи в мрежата за търсене и свързани с тях приложения, например, могат да достигнат до стотиците петабайти и хиляди възли. Те трябва да са особено издръжливи, тъй като грешките на сървъра са често срещани при такъв мащаб.

Hadoop YARN – платформа отговорна за управлението на изчислителните ресурси в клъстери и използването им за планиране на приложения на потребители.

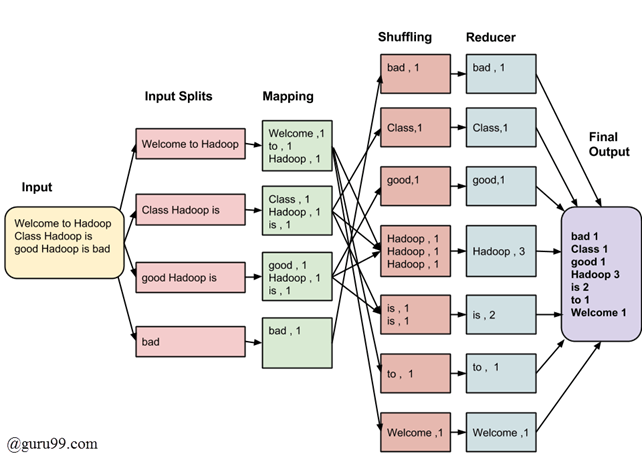
Hadoop MapReduce – реализация на модела на програмиране MapReduce за мащабна обработка на данни.

Терминът Hadoop често се използва както за базови модули и подмодули, така и за екосистема или за събиране на допълнителни софтуерни пакети, които могат да бъдат инсталирани отгоре или редом с Hadoop, като Apache Pig, Apache Hive, Apache HBase, Apache Phoenix, Apache Spark, Apache ZooKeeper, Cloudera Impala, Apache Flume, Apache Sqoop, Apache Oozie и Apache Storm.

Компонентите на MapReduce и HDFS на Apache Hadoop са вдъхновени от документите на Google за MapReduce и файловата система на Google.

Основния код на Apache Hadoop e написан най-вече на езикът за програмиране Java, малко на C и помощни програми на камандния ред написани на скриптов език. Въпреки че Hadoop MapReduce е написан на Java, всеки език може да се използва с Hadoop Streaming за реализиране на MapReduce програма.

[10] Процес на работа на Apache Hadoop MapReduce:



Данните минават през следните фази:

Разделяне на входните данни (input splits):

Входните данни в един процес на MapReduce се разделят на парчета с фиксиран размер наречени „input splits“.

„Mapping“ : Това е първата фаза от изпълнението на една “map-reduce” програма. Тук вече разделените данни се подават на т.н. „mapping“ функция, за да могат да се тя да създаде изходните стойности.

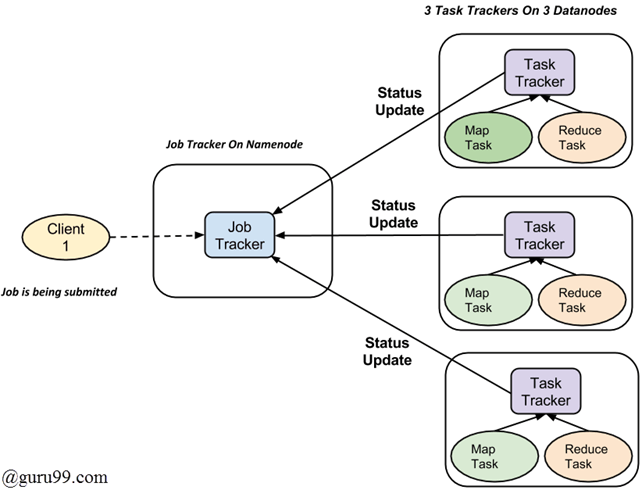
Разбъркване:

Тази фаза използва изходните данни от „mapping“ фазата. Нейната задача е да консолидира съответните записи.

Редуциране:

В тази фаза изходните стойности от фазата на разбъркване се събират. Тази фаза комбинира стойностите от разбъркващата фаза и връща единична стойност.

Диаграма на един MapReduce процес



Уеб приложението използва като главна технология Spring и Spring Boot.

Spring е технология за създаване на приложения и „inversion of control“ контейнер за Java платформата. Основните функции на технологията могат да бъдат използвани от всяко едно java приложение, но има и разширения за изграждане на уеб приложения върху платформата Java EE (Enterprise Edition).

Въпреки че технологията не налага конкретен модел на програмиране, тя стана популярна в общността на Java като допълнение или дори заместител на Enterprise JavaBeans (EJB) модела. Spring е технология с отворен код.

Spring Boot e конвенция пред конфигурация продукт на Spring за създаване на Spring базирани приложени, които с много малко конфигурации могат да започнат да работят. Едни от основните му функционалности са?

1. Създаване на самостоятелни приложения.
2. Вграден Tomcat или Jetty сървър
3. Автоматично конфигуриран Spring, когато това е възможно
4. Няма изисквания за използването на XML конфигурация

Уеб приложението използва Spring MVC за да подаде нужните данни на уеб интерфейса за визуализация.

Spring MVC e базирана на заявки технология. Тя дефинира стратегически интерфейси за всевъзможни отговорности, които трябва да бъдат обработени от една базирана на заявки технология. Целта е всеки един интерфейс да бъде прост и ясен, за може потребителите, които използват Spring MVC, да напишат своя собствена имплементация ако пожелаят. MVC проправя пътя за по-чист код в уеб интерфейса. Всички интерфейси са тясно свързани със „Servlet API“ на Spring. Това понякога затруднява ако програмистите, които използват Spring, искат да постигнат по-високо ниво на абстракция за уеб базираните приложения. Това обаче подсигурява, че всички функционалности на „Servlet API“ ще останат на разположение на програмистите, осигурявайки високо ниво на абстракция, за да улесни работата с това API.

Уеб интерфейсът е написан използвайки HTML5, CSS, JavaScript, Bootstrap и Thymeleaf.

Thymeleaf e Java XML/XHTML/HTML5  устройство за създаване на шаблони, което може да работи в уеб и не-уеб базирани среди. Той е по-подходящ за обслужване на XHTML / HTML5 в изгледния слой на уеб приложения, базирани на MVC, но може да обработва всеки XML файл дори в офлайн среда. Той осигурява пълна интеграция на Spring Framework.

В уеб приложения Thymeleaf има цел да бъде пълен заместите на JavaServer Pages (JSP) и реализира концепцията за Natural Templates: шаблонови файлове, които могат да бъдат директно отворени в браузъри и които все още се показват правилно като уеб страници.

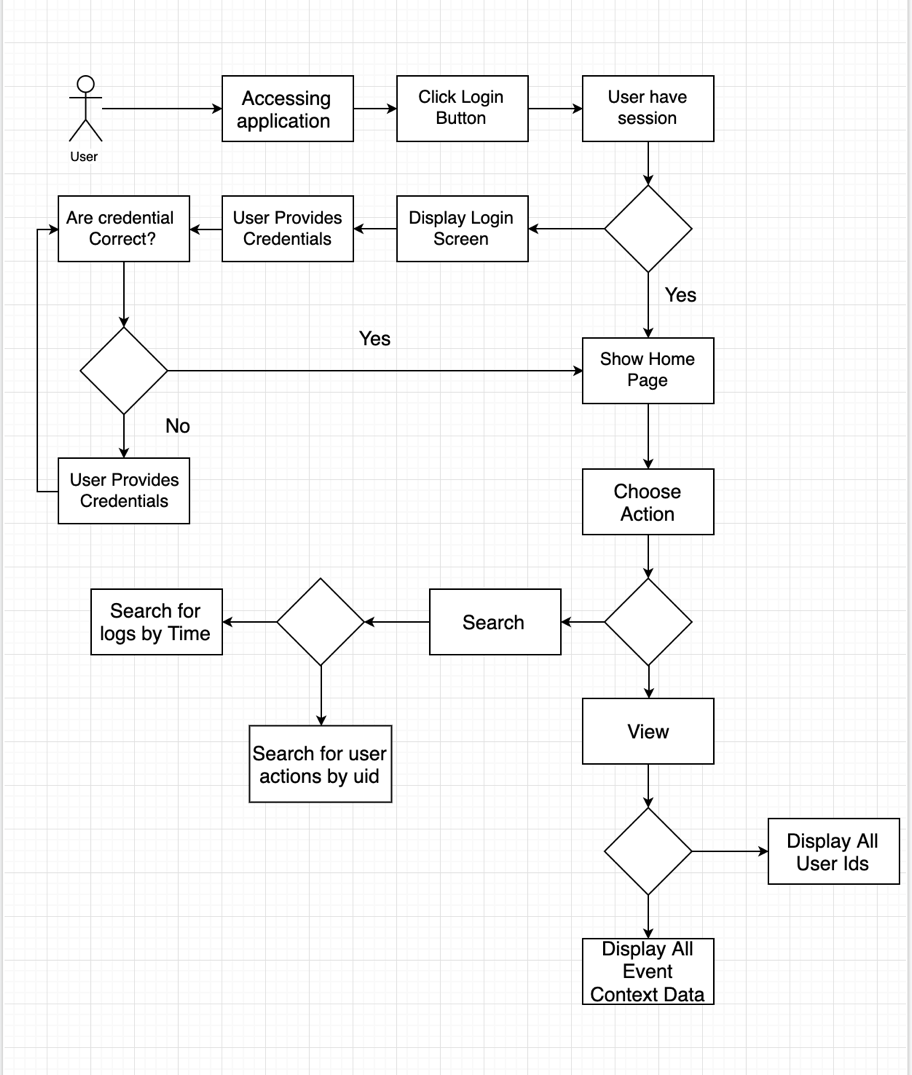
[11] Thymeleaf е софтуер с отворен код, лицензиран под лиценза Apache 2.0.

Bootstrap е технология, която се използва за дизайна на сайтове бързо и лесно. Съдържа HTML и CSS базирани шаблони за дизайн за типография, формуляри, бутони, таблици, навигация, модали, въртележки за изображения и др. Освен това ви предоставя поддръжка за JavaScript плъгини.

[12] Последният компонент е външният продукт който се използва за автентикация. Това e SAP Cloud Platform Identity Authentication Service. Той предоставя прост и сигурен облачно-базиран достъп до бизнес процеси, приложения и данни. Той опростява вашето потребителско изживяване чрез съвременни механизми за удостоверяване, сигурен „single sing-on“, интеграция с необлачни приложения. Той пази състоянието на потребителя който е извършил вход в приложението, като при затваряне на браузъра или отваряне на приложението в нова страница, не е нужно потребителят да въвежда отново своятo потребителско име или емайл и парола отново. Продуктът използва протоколът SAML2.0 за автентикация.

3. Особености на програмната реализация.

Диаграма, описва основните етапи през, които минава потребителя



Основната част на приложението или „backend“ сървъра е написана на езикът за програмиране Java.

Приложението се състои от три модула:

Модул „restApi“ където се намират моделите и контролерите използвани за осъществяване на заявки към „backend“ сървъра. Тук също се прави и обработка в подходящ формат за визуализиране от уеб интерфейса на вече обработените от „backend“ сървъра данни.

Класът Application.java и методът му „main“ служи за стартиране на приложението.

Application.java

@Controller  
@EnableAutoConfiguration  
@ComponentScan  
**public class** Application **extends** WebMvcConfigurerAdapter {

Метод чрез който се стартира приложението

**public static void** main(String[] args) {  
 SpringApplication.*run*(Application.**class**, args);  
 }

Конфигурационен метод за инициализация

@Bean  
 **public** CommandLineRunner commandLineRunner(ApplicationContext ctx) {  
 **return** args -> {  
  
 System.***out***.println(**"Let's inspect the beans provided by Spring Boot:"**);  
  
 String[] beanNames = ctx.getBeanDefinitionNames();  
 Arrays.*sort*(beanNames);  
 **for** (String beanName : beanNames) {  
 System.***out***.println(beanName);  
 }  
  
 };  
 }

Метод който пренасочва към страницата за вход:

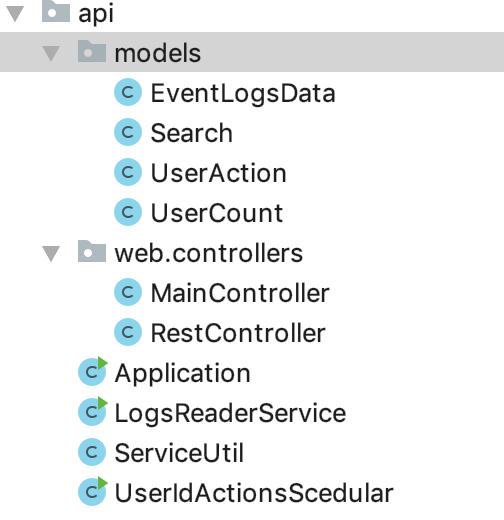
@RequestMapping(value=**"/"**)  
 **public** String welcomeGet(HttpServletResponse response) **throws** IOException {  
 **return "/login.html"**;

}

}

В директорията „/diplomna/restApi/src/main/resources“ се намират всички ресурси необходими на уеб интерфейсът като HTML, CSS и JavaScript файлове.

„Rest API“ се състои освен от „Application.java“, който служи за стартиране на приложението, тъй като то следва „Model-View-Controller“ модела. и от няколко модела, контролера и сървис класове.



MainController.java – В този клас се намират методите които свързват уеб интерфейса със „backend“ Hadoop имплементацията.

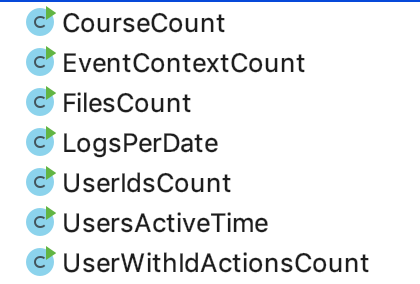
Във папката „models“ се намират всички модели които се попълват със данни и подават на уеб интерфейса са визуализация.

Двата сървис класове са ServiceUtil.java и LogsReaderService.java

Те служат за изнасяне на част от логиката която обработва изходните данни от „MapReduce“ процедурата на Hadoop.

Модул „backend“. Тук се намира кода на „backend“ сървъра. Има пакет на име „hadoop“ където могат в отделни класове могат да бъдат намерени всички имплементации на „MapReduce“ процеси, които се използва за обработка на входните данни които се намират в директорията „resources/input“. Там има файл в лог формат с размер 4.17 МБ. След като данните бъдат обработени, в зависимост какъв „MapReduce“ процес се е изпълнявал върху тях се създава поддиректория със съответното име е директорията „resources/output“

Във папката „/diplomna/backend/src/main/java/hadoop“ се намират класовете, които правят обработката на лог файла използвайки Apache Hadoop библиотеки, които стартират „MapReduce“ процесите.



В тези класове се намира имплементацията на програми която стартират „MapReduce“ процеси.

Тази имплементация включва наследяването на два класа Mapper и Reducer.

Вътрешният клас TokenizerMapper.java наследява Mapper класът. Този клас и методът му map служи за филтриране на данните в зависимост от това дали те изпълват желаните изисквания. В случая на UserWithIdActionsCount.java класът редът от текст да съдържа текста "The user with id '" + userId + "'", като „userId“ е подаденият от „rest“ сървъра потребителски идентификатор. Ако това е така пише в  „context“ променливата неговата стойност и това колко пъти се среща, в случая е един път. Така докато свършат всички редове във файла.

Вътрешният клас Reducer и методът му „reduce“ получава списък от обработените от Mapper класът и ги сумира броят на елементите, които имат една и съща стойност.

В методът „ execute“ са имплементирани конфигурациите които са нужни на Hadoop да стартира „MapReduce“ процес.

Дава се име на самия процес, определя се кои ще са „mapper“ и „reducer“ класовете, кой ще е класът от който ще се стартира процесът, директории за изходни и входни данни.

UserWithIdActionsCount.java

**public class** UserWithIdActionsCount {  
   
 **private static** String *userId*;  
  
 **public static class** TokenizerMapper  
 **extends** Mapper<Object, Text, Text, IntWritable> {  
  
 **private final static** IntWritable ***one*** = **new** IntWritable(1);  
 **private** Text **word** = **new** Text();  
  
 **public void** map(Object key, Text value, Context context  
 ) **throws** IOException, InterruptedException {  
 String pattern = **"The user with id '"** + *userId* + **"'"**;  
  
 Pattern r = Pattern.*compile*(pattern);  
 Matcher m = r.matcher(value.toString());  
  
 File file = **new** File(**"/Users/i338442/diplomna/backend/src/main/resources/userIdPlusDateCount"** + *userId* + **".txt"**);  
 file.createNewFile();  
  
 **try**(  
 FileWriter fw = **new** FileWriter(**"/Users/i338442/diplomna/backend/src/main/resources/userIdPlusDateCount"** + *userId* + **".txt"**, **true**);  
 BufferedWriter bw = **new** BufferedWriter(fw)) {  
  
 **if** (m.find()) {  
 String[] split = value.toString().split(**","**);  
   
 bw.write(split[0] + **","** + split[1]);  
 bw.newLine();  
 String userAction = split[5];  
 context.write(**new** Text(userAction), **new** IntWritable(1));  
 }  
   
 }  
  
 }  
  
 }  
  
 **public static class** IntSumReducer  
 **extends** Reducer<Text, IntWritable, Text, IntWritable> {  
 **private** IntWritable **result** = **new** IntWritable();  
  
 **public void** reduce(Text key, Iterable<IntWritable> values,  
 Context context  
 ) **throws** IOException, InterruptedException {  
 **int** sum = 0;  
 **for** (IntWritable val : values) {  
 sum += val.get();  
 }  
 **result**.set(sum);  
 context.write(key, **result**);  
 }  
 }

**public static void** execute(String[] args) **throws** IOException, ClassNotFoundException, InterruptedException {  
 *userId* = args[0];  
 Configuration conf = **new** Configuration();  
 Job job = Job.*getInstance*(conf, **"UserIds actions Count"** + *userId*);  
 job.setJarByClass(UserWithIdActionsCount.**class**);  
 job.setMapperClass(TokenizerMapper.**class**);  
 job.setCombinerClass(IntSumReducer.**class**);  
 job.setReducerClass(IntSumReducer.**class**);  
 job.setOutputKeyClass(Text.**class**);  
 job.setOutputValueClass(IntWritable.**class**);  
 FileInputFormat.*addInputPath*(job, **new** Path(**"/Users/i338442/diplomna/backend/src/main/resources/input"**));  
  
 FileOutputFormat.*setOutputPath*(job, **new** Path(**"/Users/i338442/diplomna/backend/src/main/resources/outputUserIdsActionsCount"** + *userId*));  
 **boolean** b = job.waitForCompletion(**true**);}  
}

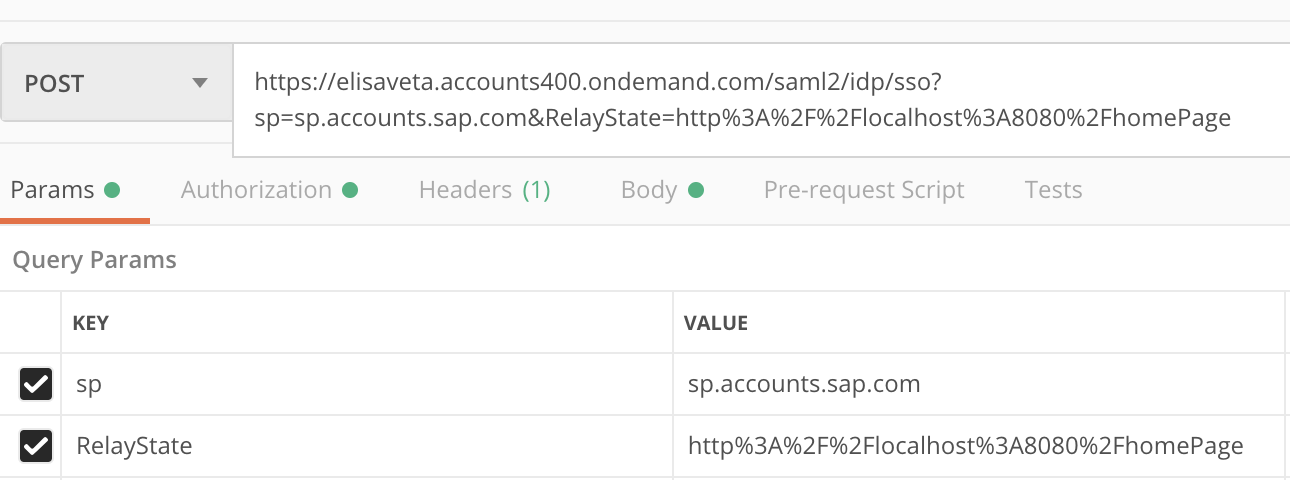
Другите класове са с подобна имплементация и техният код може да бъде намерен в GitHub, като пак дефинират същите методи и същите вътрешни класове с тази разлика че методът „map“, който се намира във вътрешния клас „TokenizerMapper“. Също така се конфигурират различни директории за изходните данни.

Това става тук:

FileOutputFormat.*setOutputPath*(job, **new** Path(**"/Users/i338442/diplomna/backend/src/main/resources/outputUserIdsActionsCount"** + *userId*));

Той се използва от другите два модула. Целта му е да форматира вече обработените от Hadoop потребителски идентификатори във формат удобен за ползване. Съдържа един клас UsersIdFormatter.java

Приложението е изцяло написано на Java, като за уеб интерфейсът се интерфейсът са използвани HTML5, CSS и JavaScript. То използва Spring boot като стартиращ „engine“, и отговаря на адрес <http://localhost:8080>. Щом този адрес бъде достъпен потребителят бива пренасочен към входната страница на приложението където се намира голям син бутон с надпис „Login“ на него. Кликайки на този бутон се изпълнява следната заявка към сървърът за автентикация:



Във „RelayState“ параметъра на заявката е посочена началната страница на приложението, към която потребителят ще бъде пренасочен след успешна автентикация.

След изпълнението на тази заявка се появява екранът за вход на сървърът за автентикация. След като потребителят въведе правилните данни за автентикация той бива пренасочен към адресът зададен в „RelayState“ параметър на заявката описана по-горе.

След като потребителя бъде пренасочен към началната страница на приложението уеб интерфейсът изпълнява следните заявки към „backend“ сървъра за да получи данните които са му необходими за визуализиране.



Който връща отговор в html format:



Предварително е стартирани „MapReduce“ процедурите за:

1. „UserIdsCount“ - намира всички идентификационни номера на потрбители.
2. „UsersActiveTime“ – намира количеството логове за месец за 2018 година.
3. „CourseCount“ – намира броят на курсовете
4. „EventContextCount“ – намира типът на събитията и колко често те се срещат.

Данните се подават на уеб интерфейса в json формат като биват констроирани в следния формат:

{

"canvasData" : “[1, 2, 3, 4]”,

“mostActiveLog” : 3535,

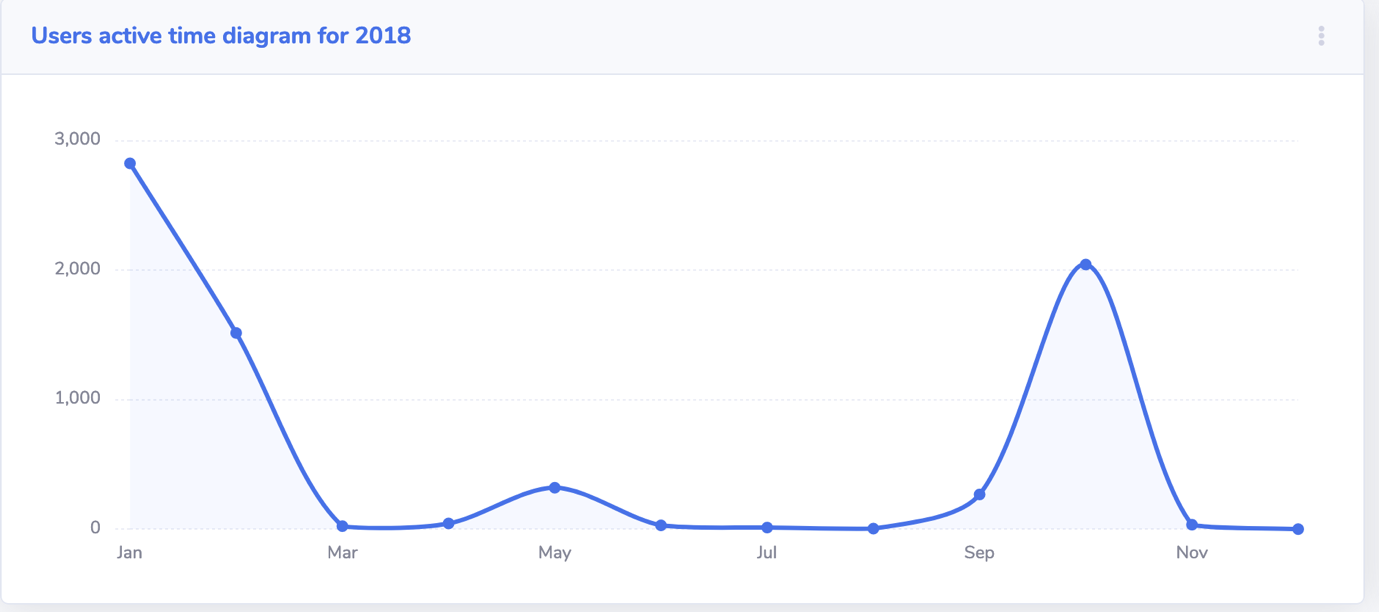
“totalOfLogs”: 736474,

“totalOfUsers”: 646,

“numberOfCourses”: 1,

“pieCharData”: “[1, 2, 3, 4]”  
}

"canvasData" – това са данните за броят на логове, групирани по месец и се визуализира в тази графика



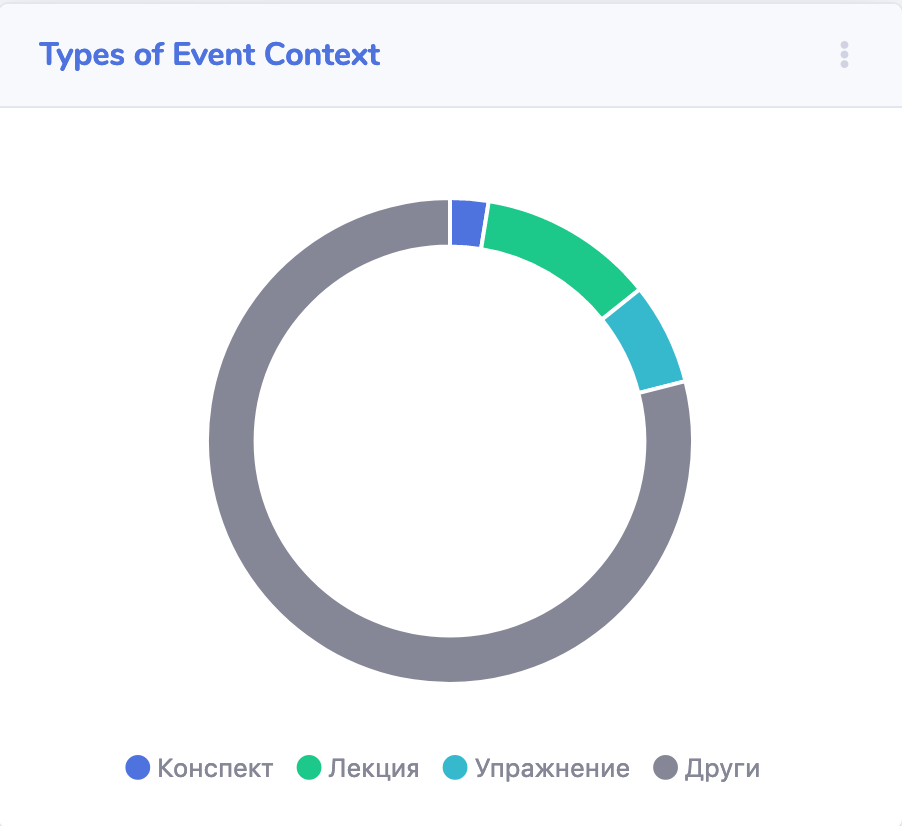
“mostActiveLog” – броят на логове в месецът с най-много логове

“totalOfLogs” – сумата на всички логове за 2018 година

“totalOfUsers” – броят на всички потребители

“numberOfCourses” – борят на всички курсове

“pieCharData” – данните за броят на типовете събития, използва се за тази графика



За да се намери стойността на "canvasData" се взема резултатът от „UsersActiveTime“ процедурата, филтрират се данните за да се изкарат само тези които са за 2018 година, и се връща списък от цели числа.

За да се намери стойността на “mostActiveLog” вече филтрираните данни които се попълват е "canvasData" се сортират в нарастващ ред и се взема последният елемент, който е с най-голяма стойност.

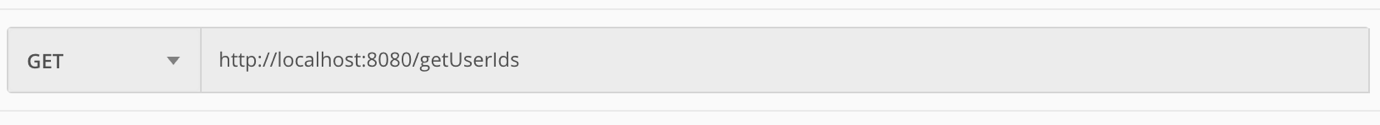
“totalOfUsers” - взема резултатът от „UserIdsCount“ процедурата и се сумират всички потребителски идентификатори.

“numberOfCourses” - взема резултатът от „CourseCount“ процедурата и се сумират всички курсове.

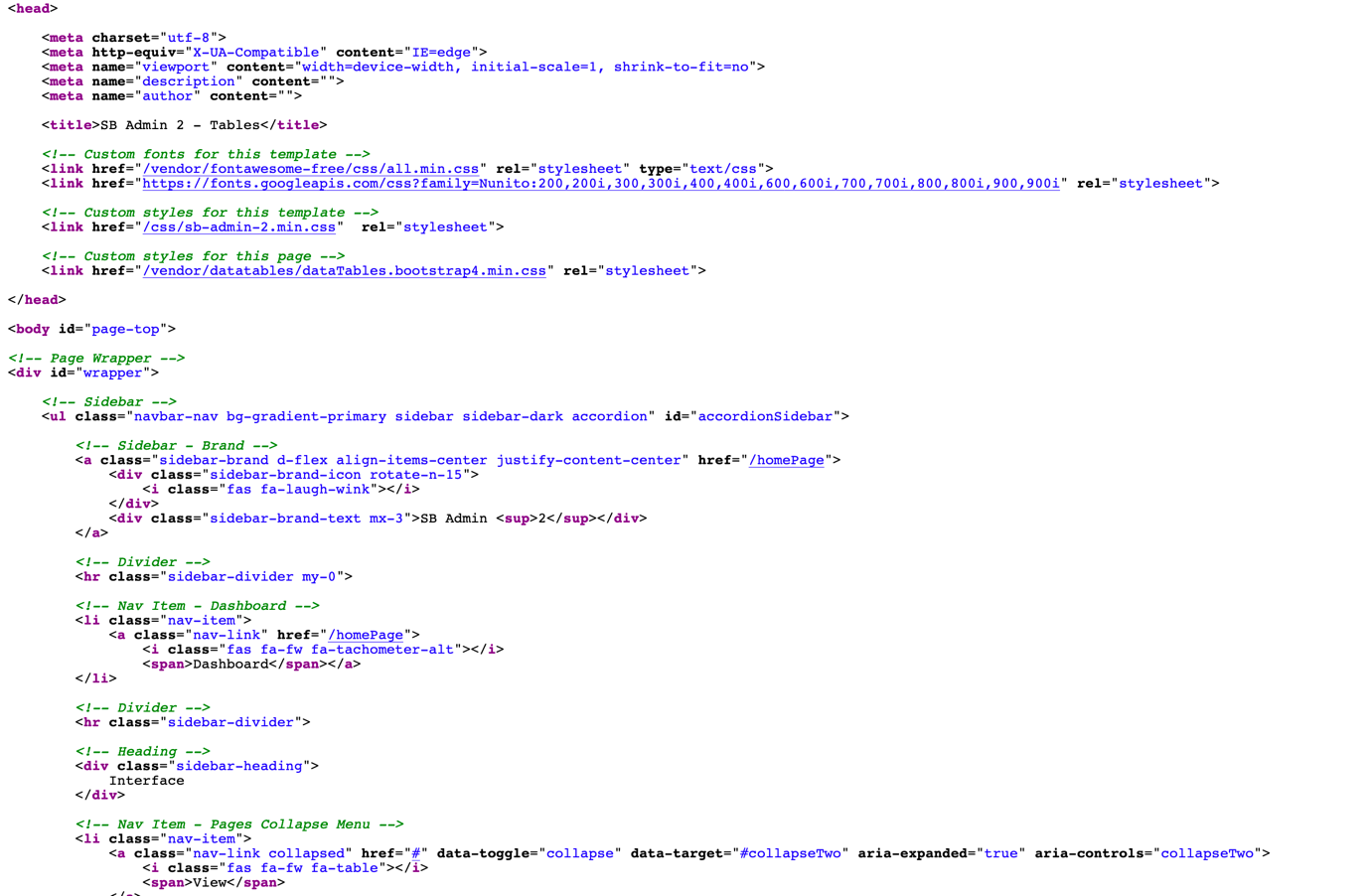
„pieCharData“ – взема резултатът от „EventContextCount“ процедурата, данните се групират по типове и се намира колко пъти всеки тип се среща в логовете.

Потребителят може да избере между преглед на резултатът на вече обработени данни или да подаде заявка за стартиране на обработка на данни.

При преглед на „All user Ids“ се изпълнява следната заявка към „backend“ сървъра



Която връща отговор в HTML формат



За да се попълнят данните в таблицата на уеб интерфейсът се взема резултатът от „UserIdsCount“  процедурата.

Данните се обработват и се конструира списък от данни на „userActions“ модела

{

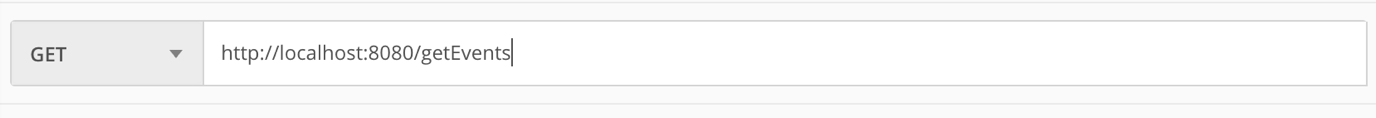
“name”: “2”,

“count”: 4

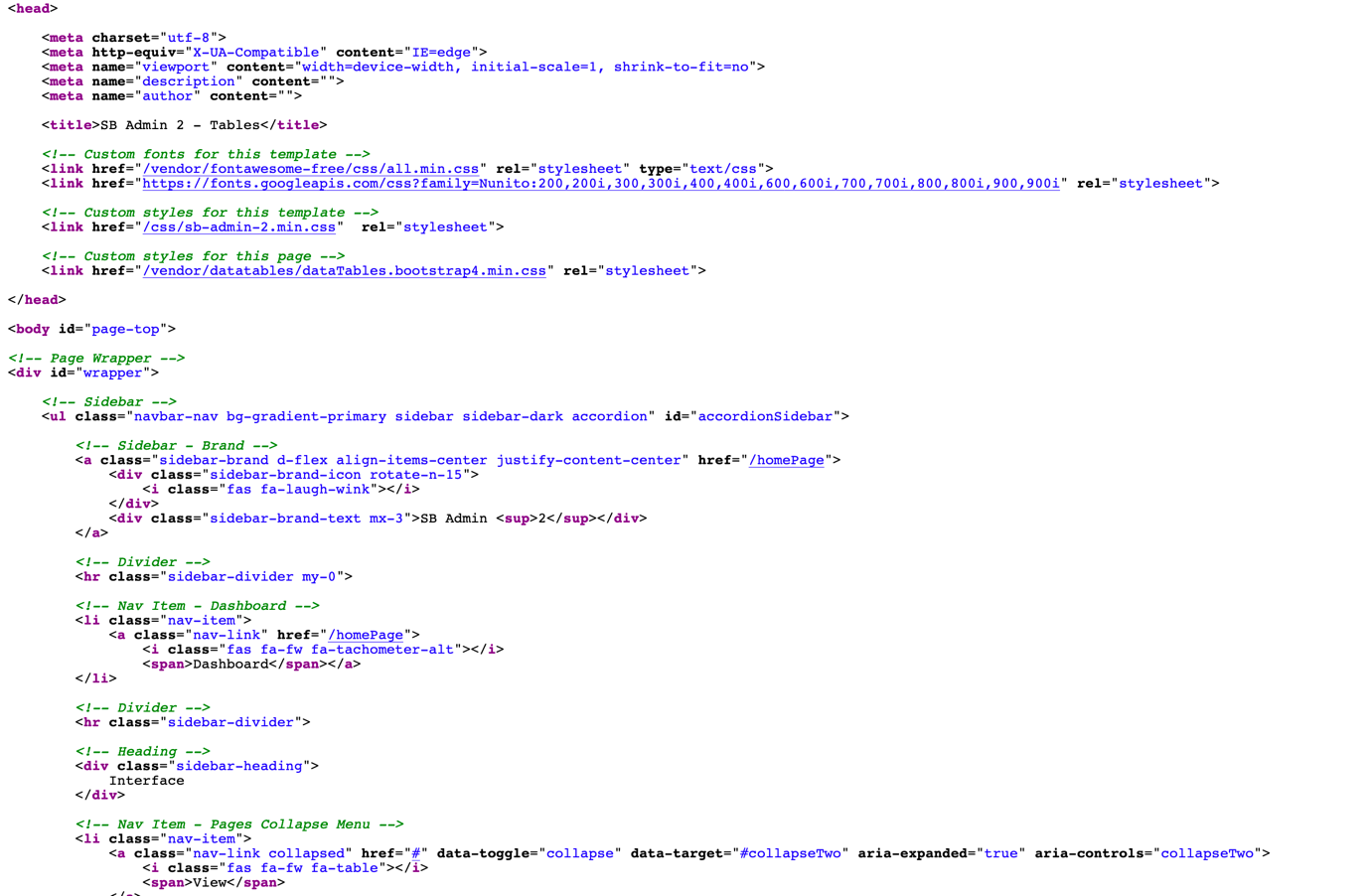
}

Тези данни се подават на tablе.html шаблонът, където биват визуализирани.

При преглед на „Types of events“ се изпълнява следната заявка към „backend“ сървъра



Която връща отговор в HTML формат



За да се попълнят данните в таблицата на уеб интерфейсът се взема резултатът от „EventContextCount“  процедурата.

Данните се обработват и се конструира списък от данни на „userActions“ модела

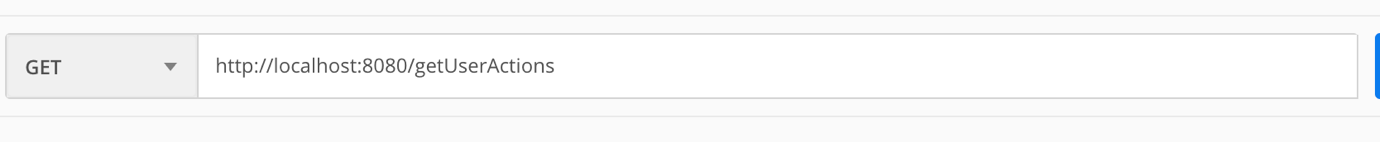
{

“name”: “File: Software contractions”,

“count”: 246

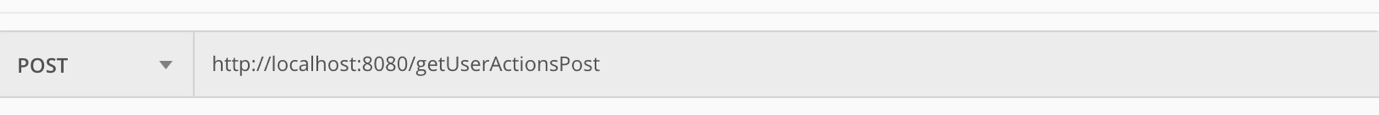
}

Ако потребителят избера да търси „User actions by user id“ се изпълнява следната заявка към „backend“ сървъра



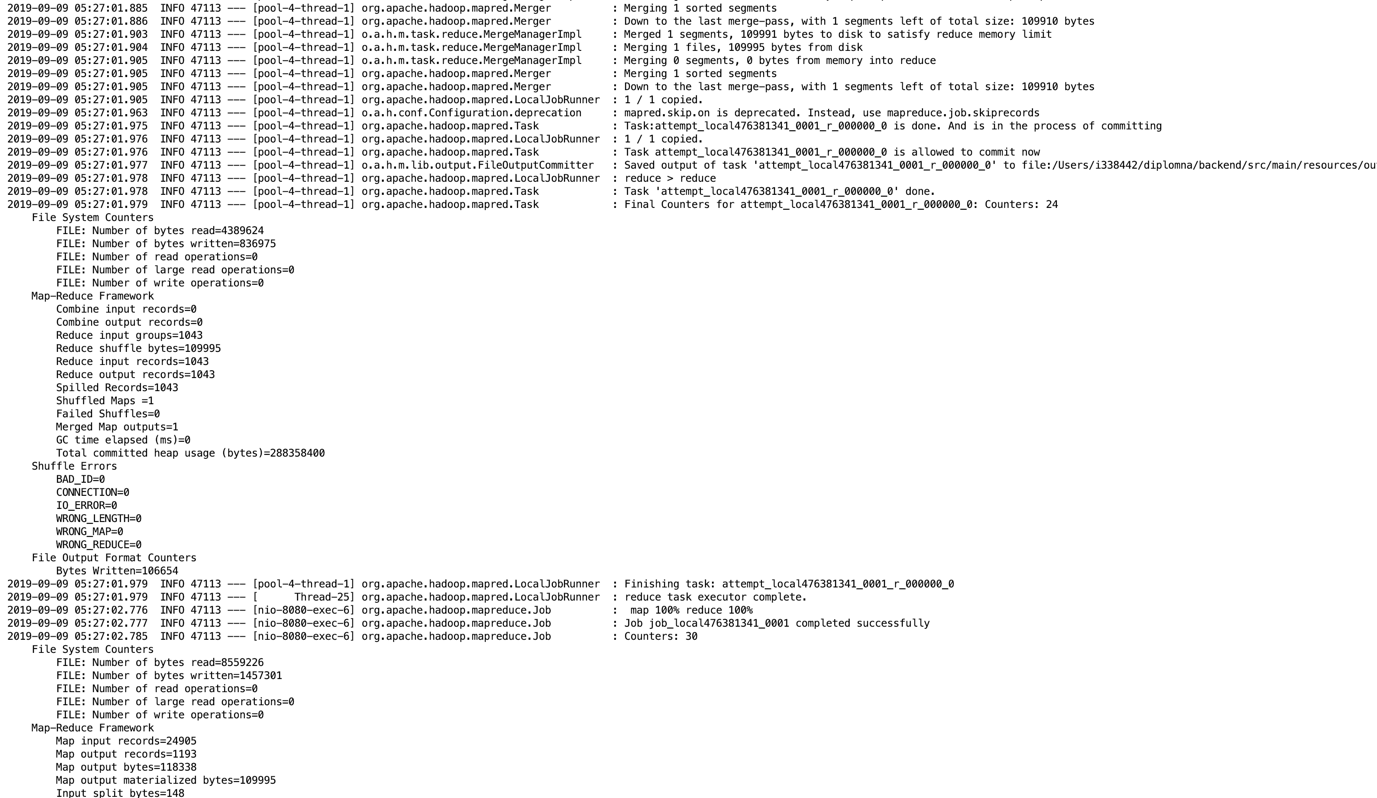
Връща се отговор в HTML формат с празна таблица.

Когато потребителят въведе потребителски идентификатор за когото иска да види какви действия е извършвал, в полето за търсене той въвежда потребителския идентификатор. След като натисне клавишът “Enter” се изпълнява следната заявка към „backend“ сървъра



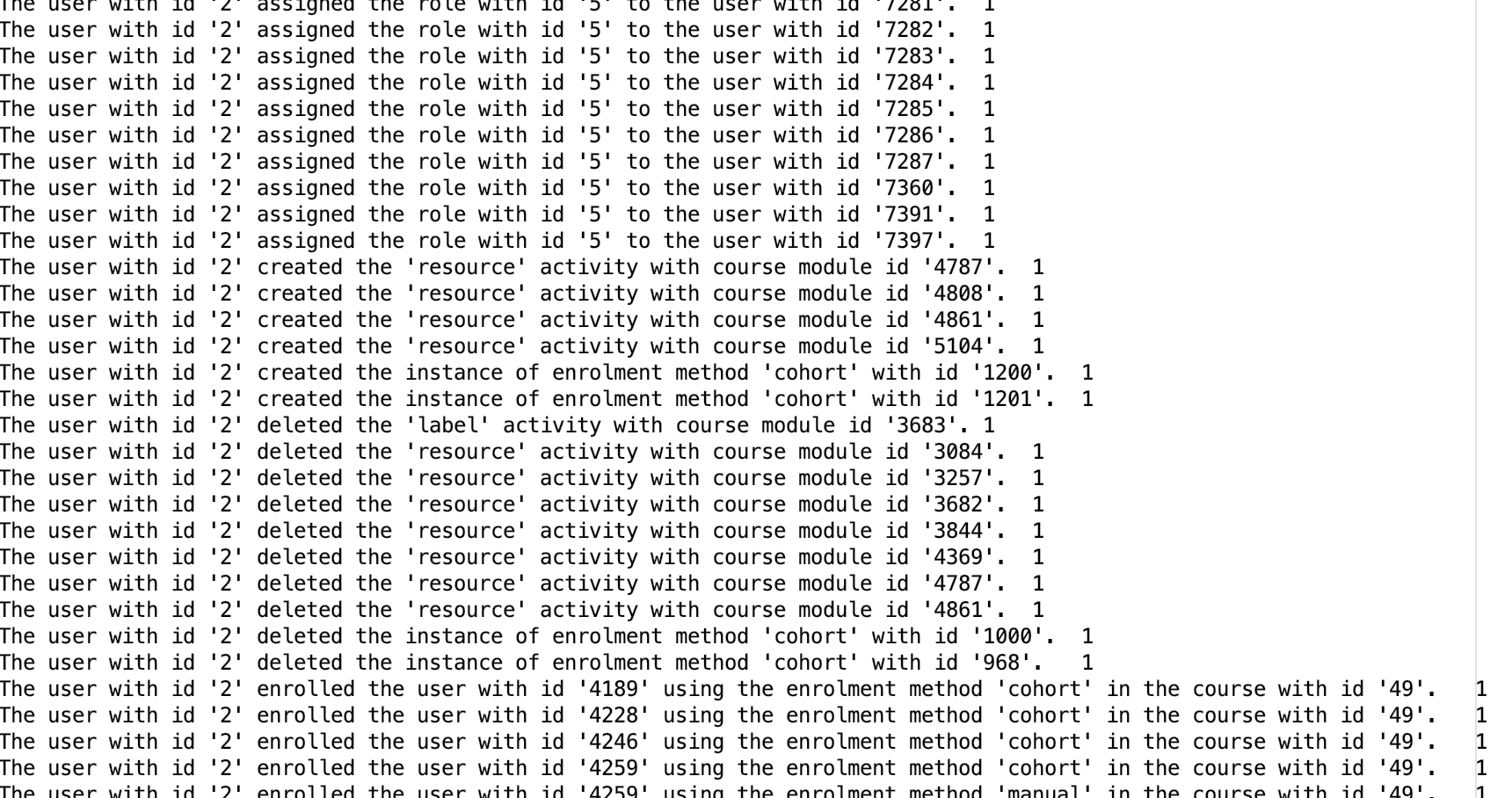
Връща се отговор в HTML формат с попълнена таблица.

Преди да бъдат върнати данните които ще се визуализират от уеб интерфейсът, ако те вече не да били обработени ще се стартира “MapReduce” процедура от тип „UserWithIdActionsCount“ за намиране на данните за подадения потребителски индентификатор.



Това може да отнеме различно време в зависимост от това колко данни има за този потребител.

Примерен изход от “MapReduce” процедурата



След това данните се обработват и се конструира списък от данни на „userActions“ модела.

{

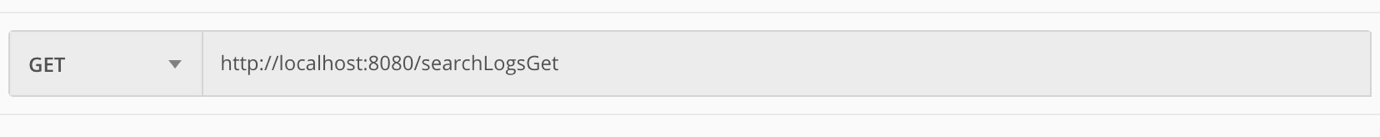
“name”: “The user with id '2' assigned the role with id '5' to the user with id '4309'.”,

“count”: 1

}

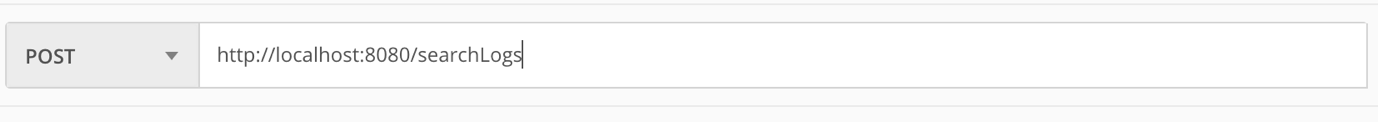
Подават се на table.html шаблонът който ги попълва и визуализира.

Ако потребителят избера да търси „Logs by Time“ се изпълнява следната заявка към „backend“ сървъра.



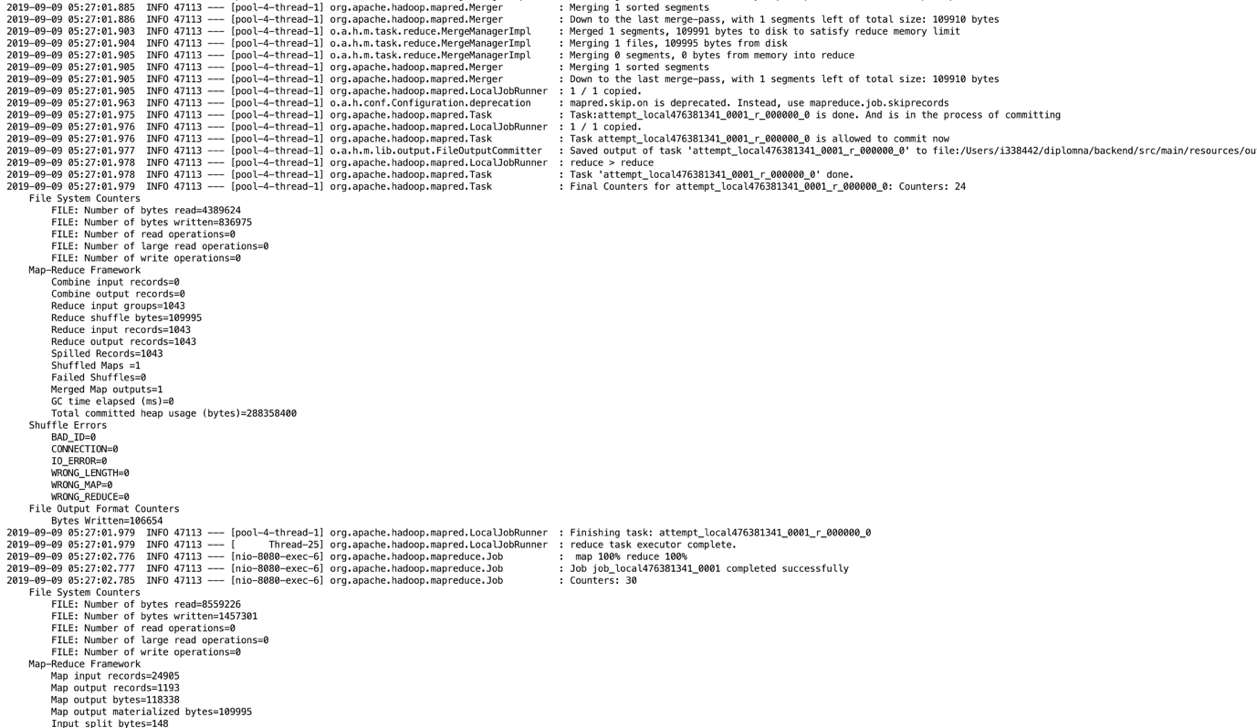
Връща се отговор в HTML формат с празна таблица.

Когато потребителят въведе дата във „d/mm/yy“ формат за която иска да види всички действия извършени на тази дата, в полето за търсене той въвежда датата. След като натисне клавишът “Enter” се изпълнява следната заявка към „backend“ сървъра.



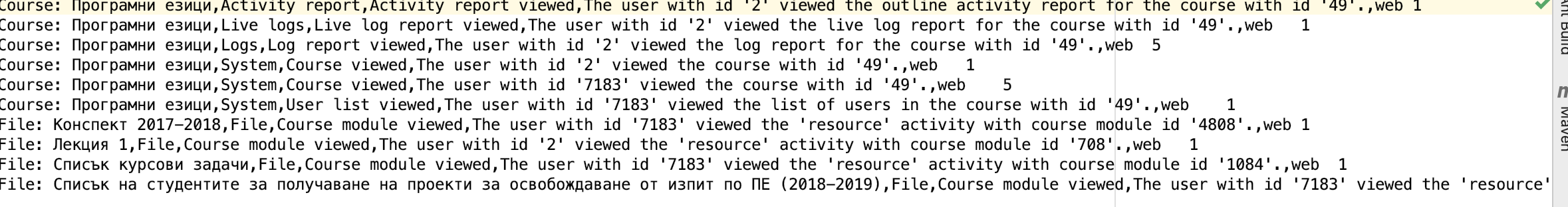
Връща се отговор в HTML формат с попълнена таблица.

Преди да бъдат върнати данните които ще се визуализират от уеб интерфейсът, ако те вече не да били обработени ще се стартира “MapReduce” процедура от тип „LogsPerDate“ за намиране на данните за подаденото време.



Това може да отнеме различно време в зависимост от това колко данни има за този дата.

Примерен изход от “MapReduce” процедурата



След това данните се обработват и се конструира списък от данни на „EventLogsData“ модела.

{

“count” 1,

“eventContext” : “Course: Програмни езици”,

“component” : “Live logs”,

“eventName”: “Live log report viewed”,

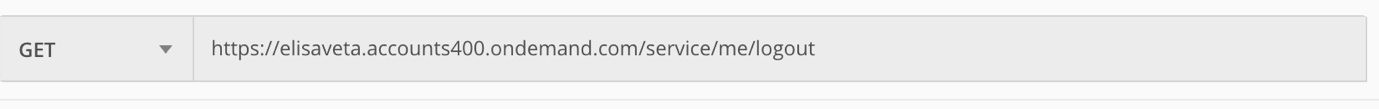
“eventDescription” : “The user with id '2' viewed the live log report for the course with id '49'”

}

Подават се на table.html шаблонът който ги попълва и визуализира.

При желание потребителят може да приключи своята сесия като излезе от приложението.

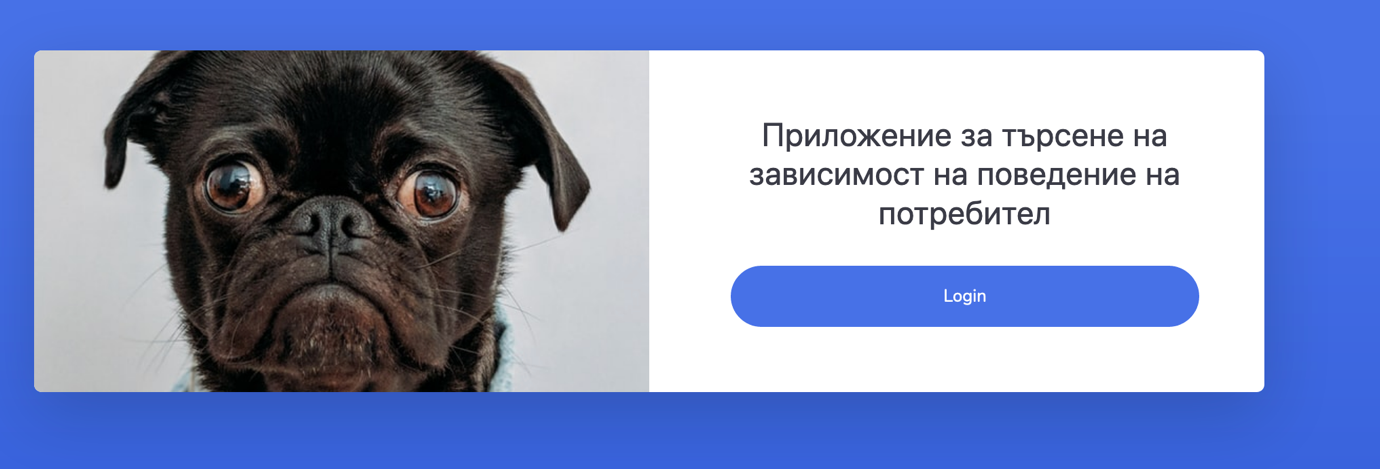
Излизайки от приложението се праща следната заявка към сървъра за автентикация



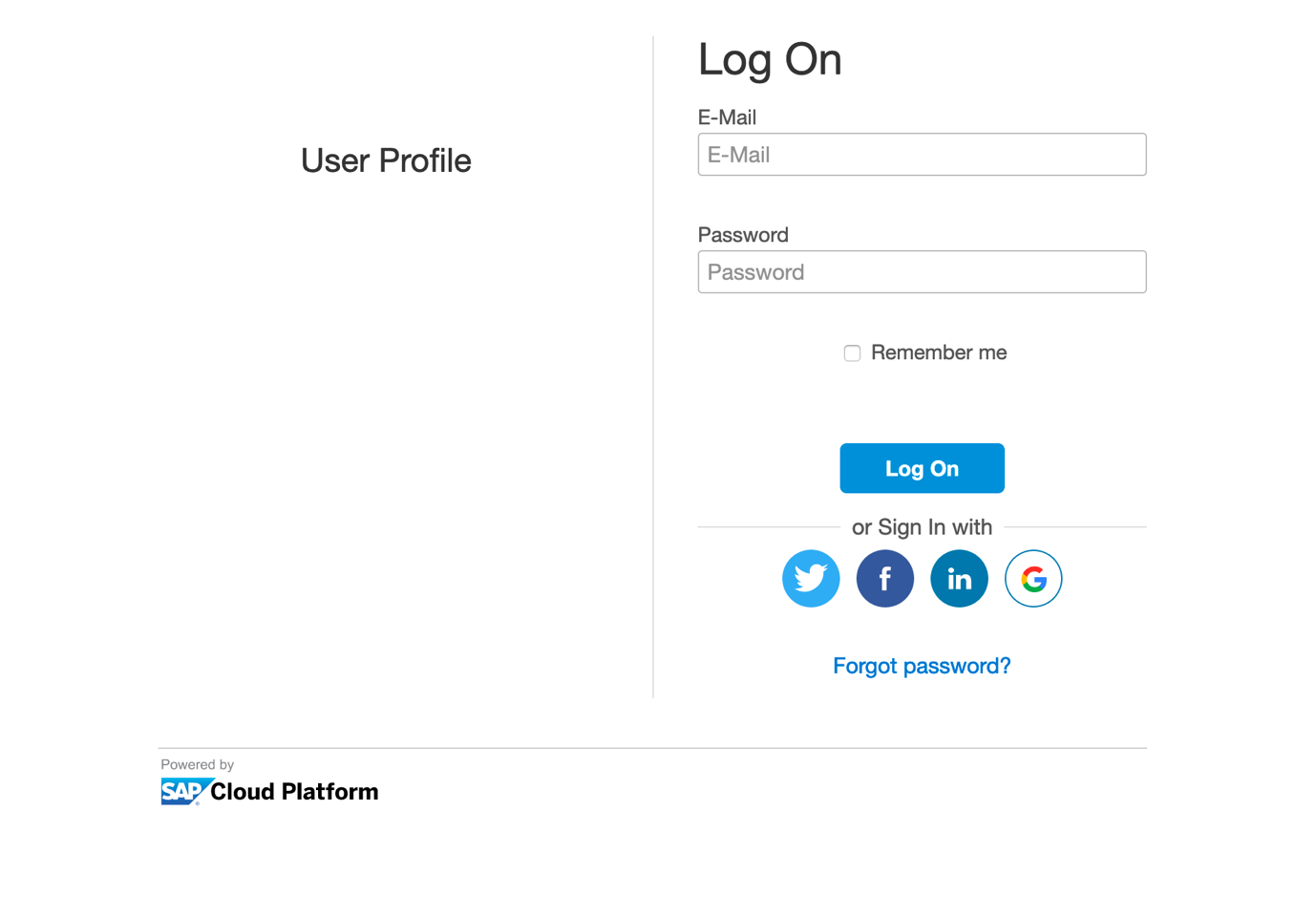
Сървърът връща статус 200 ОК и зачиства всички бисквитки на сесията от браузъра.

4. Ръководство за потребителя

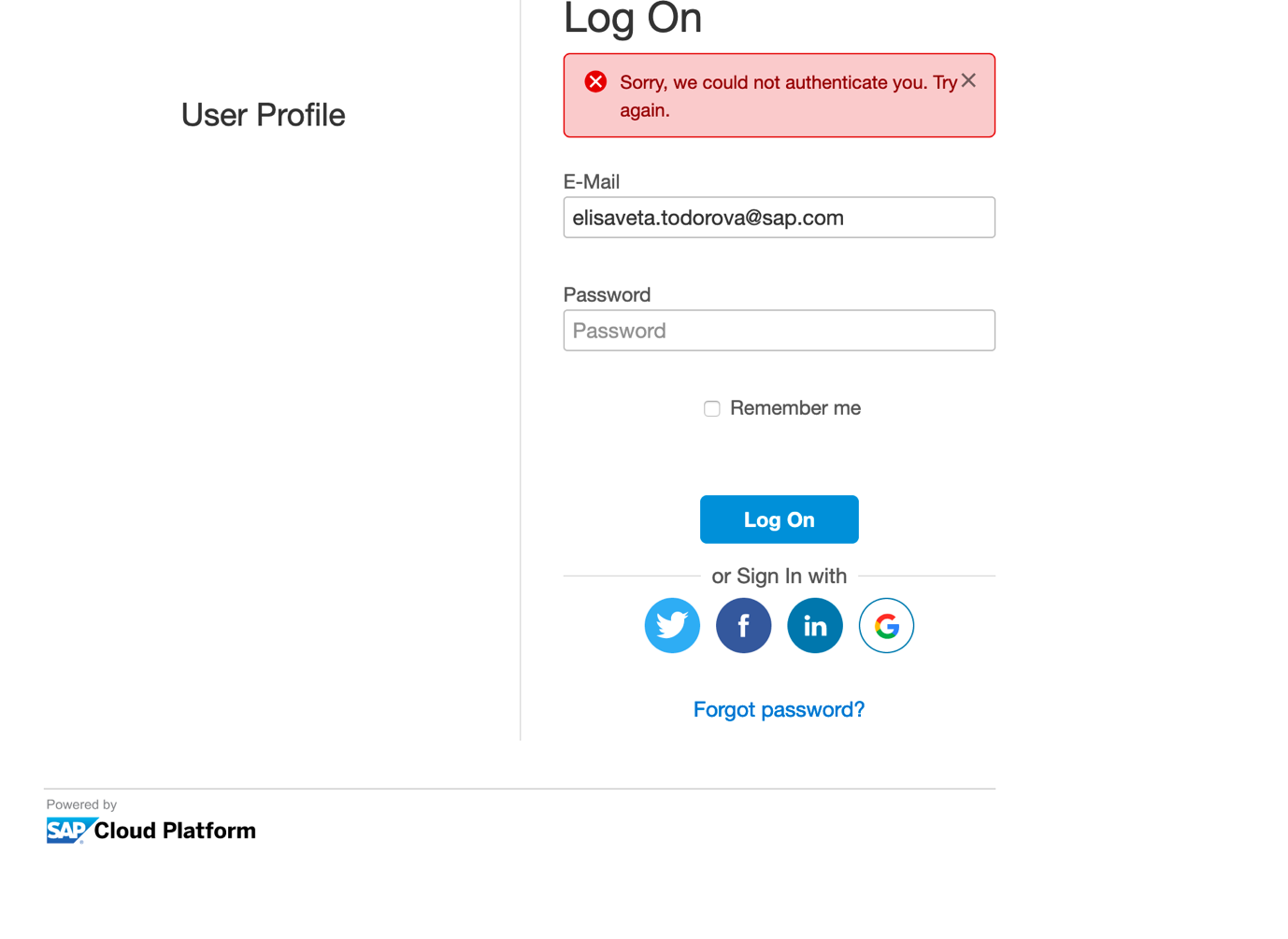
При достъпване на приложението на адрес <http://localhost:8080> потребителят бива пренасочен към екранът за вход на приложението.



Кликайки върху синият бутон с надпис „Login“, ако потребителят няма вече активна сесия в текущи браузър той бива пренасочен към страницата за вход на сървъра за автентикация.

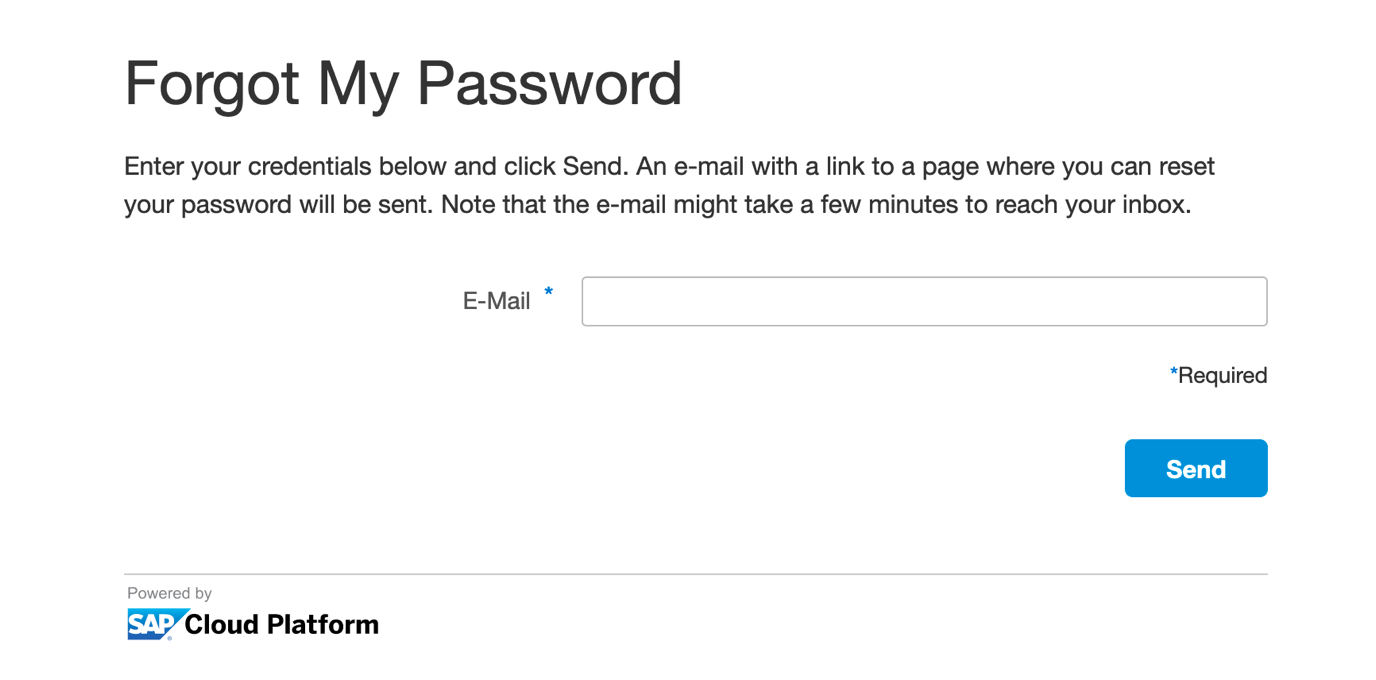


Ако потребителят сбърка емайла или паролата си ще му излезе следния екран с надпис „Sorry, we could not authenticate you. Try again.“

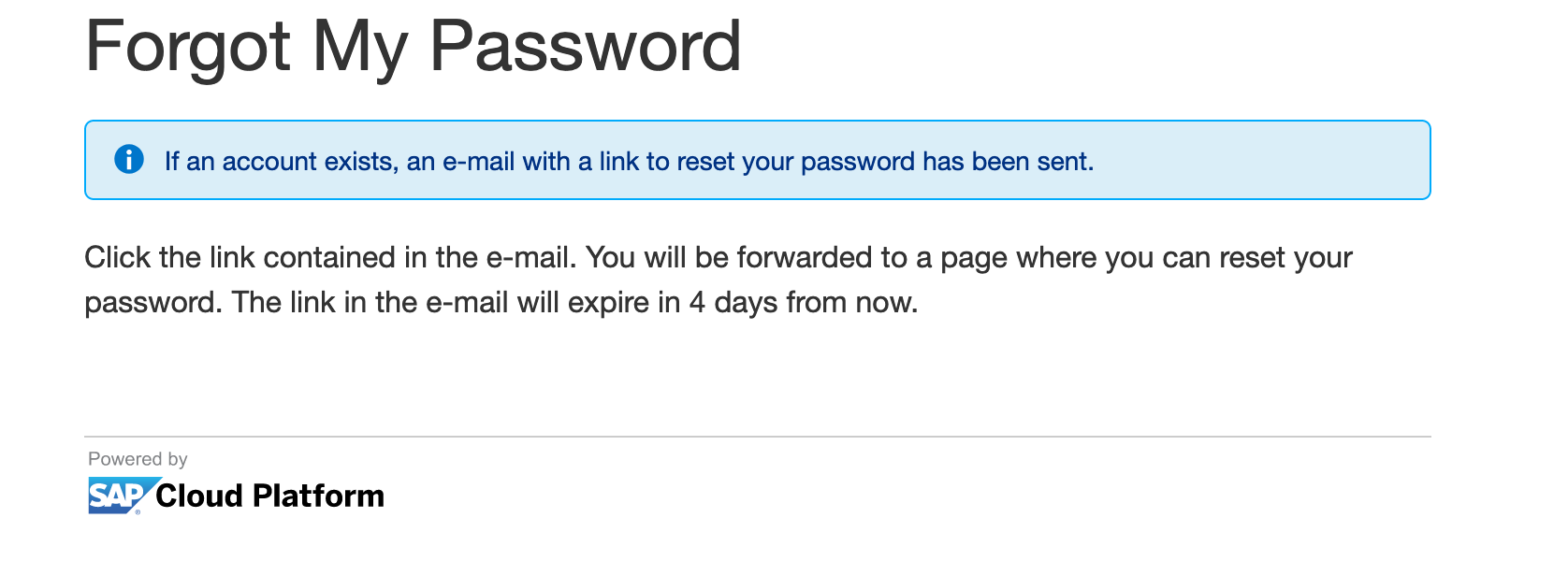


При забравена парола потребителят може да я смени, като кликне на линка с текст „Forgot password?“.

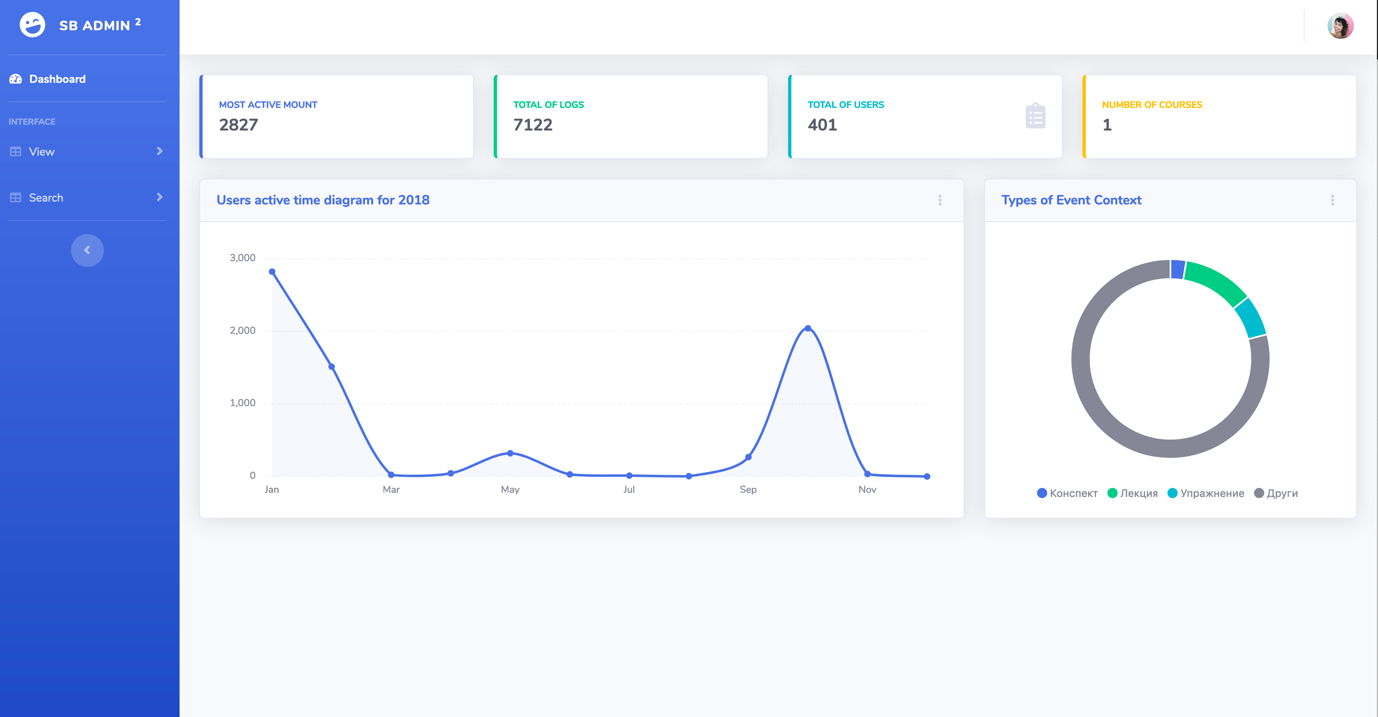
Той бива пренасочен към екран на който трябва да въведе своя емайл адрес.



След въвеждане на емайл и кликане на бутона с надпис „Send“, потребителят бива пренасочен към страница която пояснява, че емайл за смяна ма парола ще бъде изпратен, само ако потребител с такъв емайл адрес съществува.

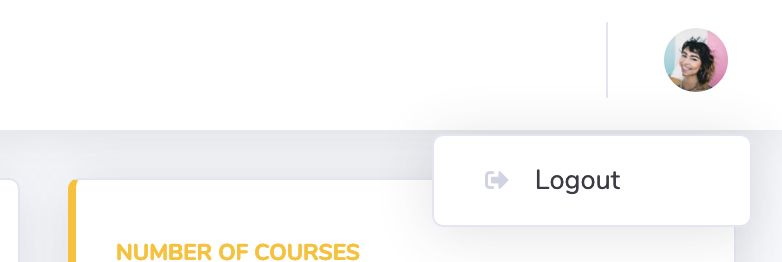


Ако потребителят успешно въведе своя емайл адрес или парола то бива пренасочен към началната страница на приложението.

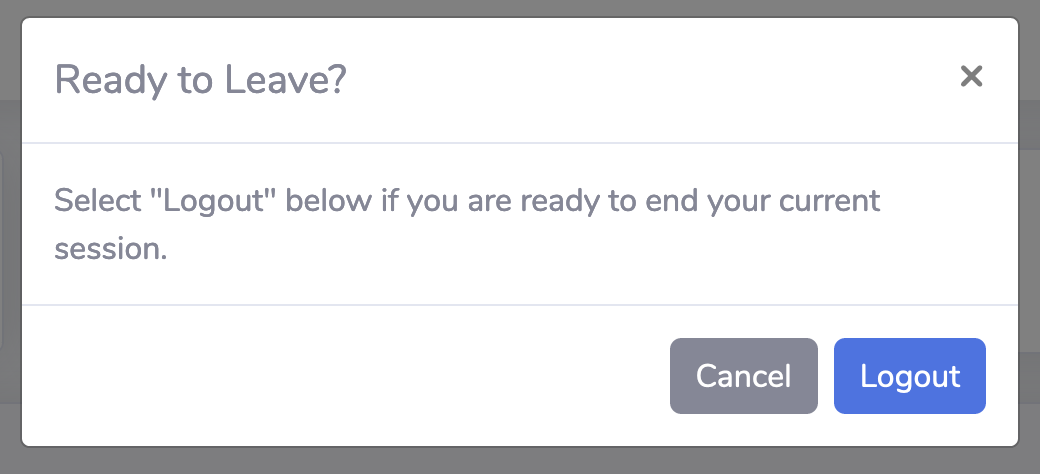


В десния горен ъгъл се иконката на логнатия потребител.

Като се кликне върху нея се появява меню с опция за „Logout“



Кликайки на бутона „Logout“ се изписва следното съобщение

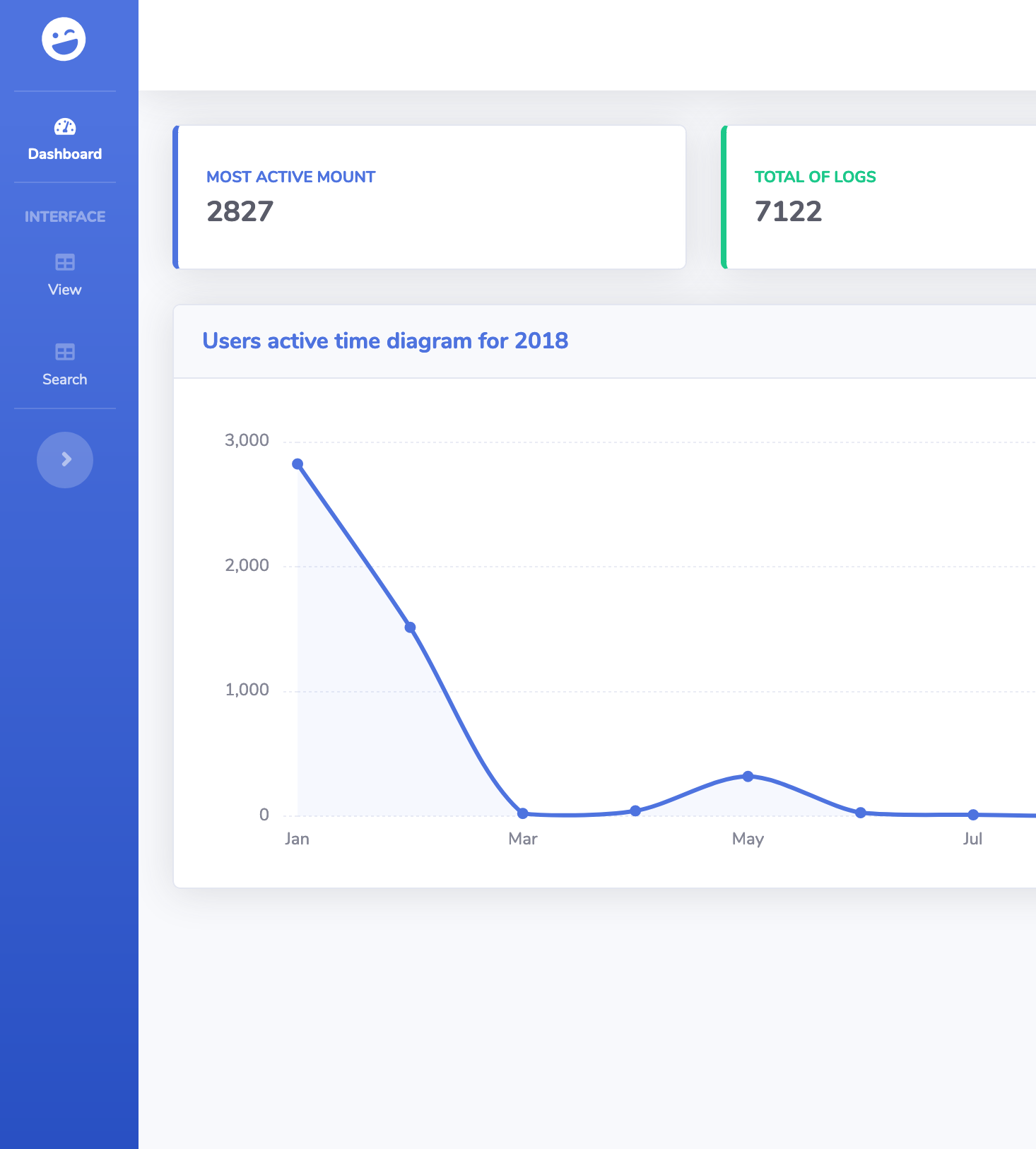


Натискайки бутона „Cancel“ потребителят остава на същата страница.

Натискайки бутона „Logout“ се убива потребителската сесия и потребителят бива пренасочен към празна страница.

Менюто от страни може да се свива ако приложението се използва на по-малък екран. Това става чрез кликане на стрелкат оградена в кръг.

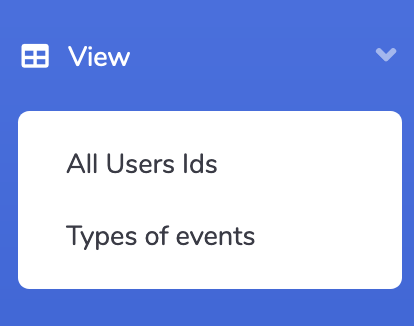




В синьото меню отляво се виждат компонента, две от които са менюта, а трето „Dashboard“ избирането му потребителят е пренасочен към началната страница на приложението.



При кликне върху текста с надпис „View“  се появява подменю с две опции.



Избирайки опцията „All Users Ids“ потребителят бива пренасочен към страница с таблица съдържаща всички потребителски идентификатори и това колко често се срещат



Най-отдолу на таблицата са изобразени колко са записите и колко от тях се визуализират на текущата страница, също така има и бутони за преминаване на следваща или предишна страница.

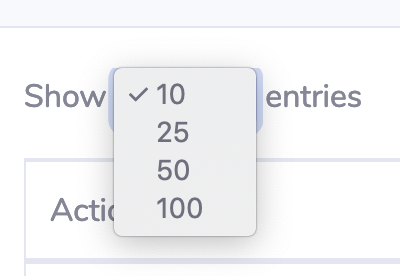


Потребителят има възможността да избере колко от тези записи да бъдат визуализиране на страницата.

Като кликне на тази стрелка

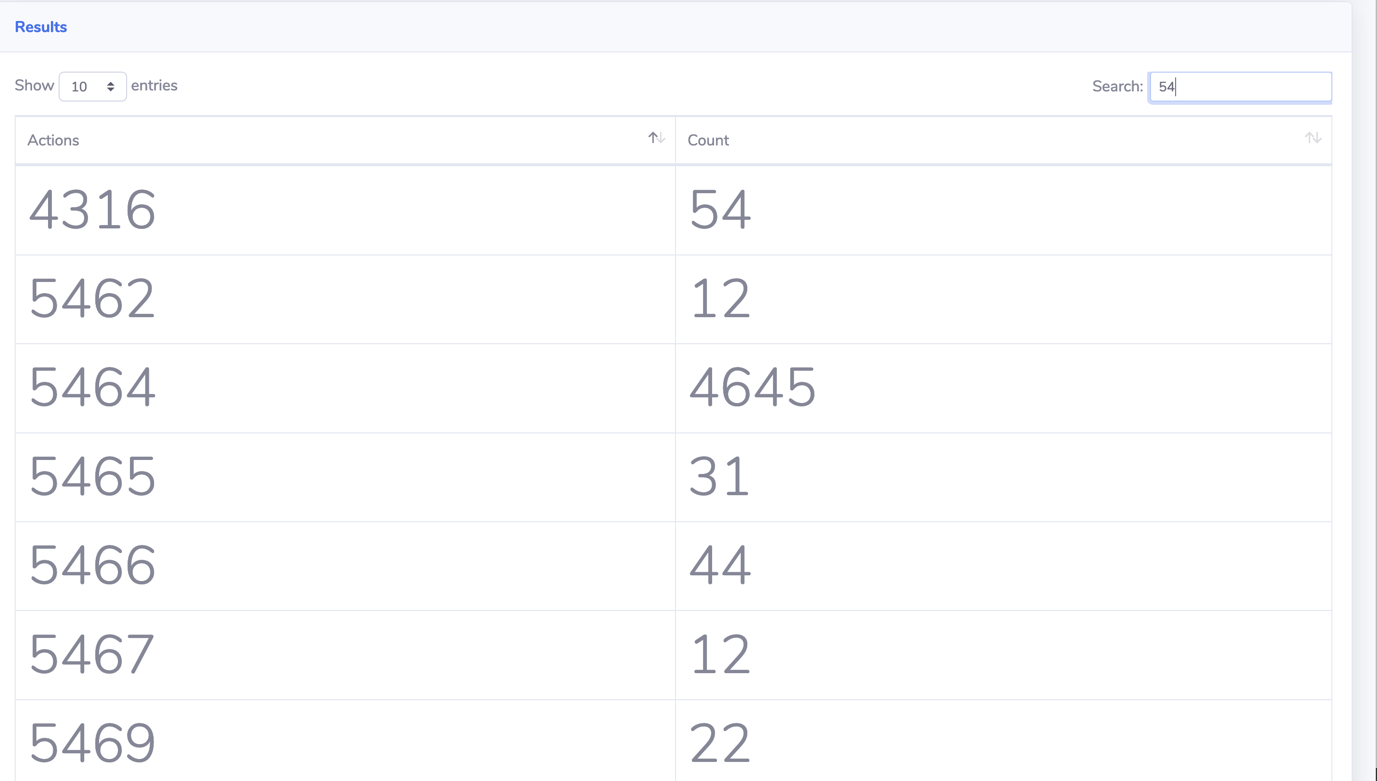


Това меню с опции се появява.

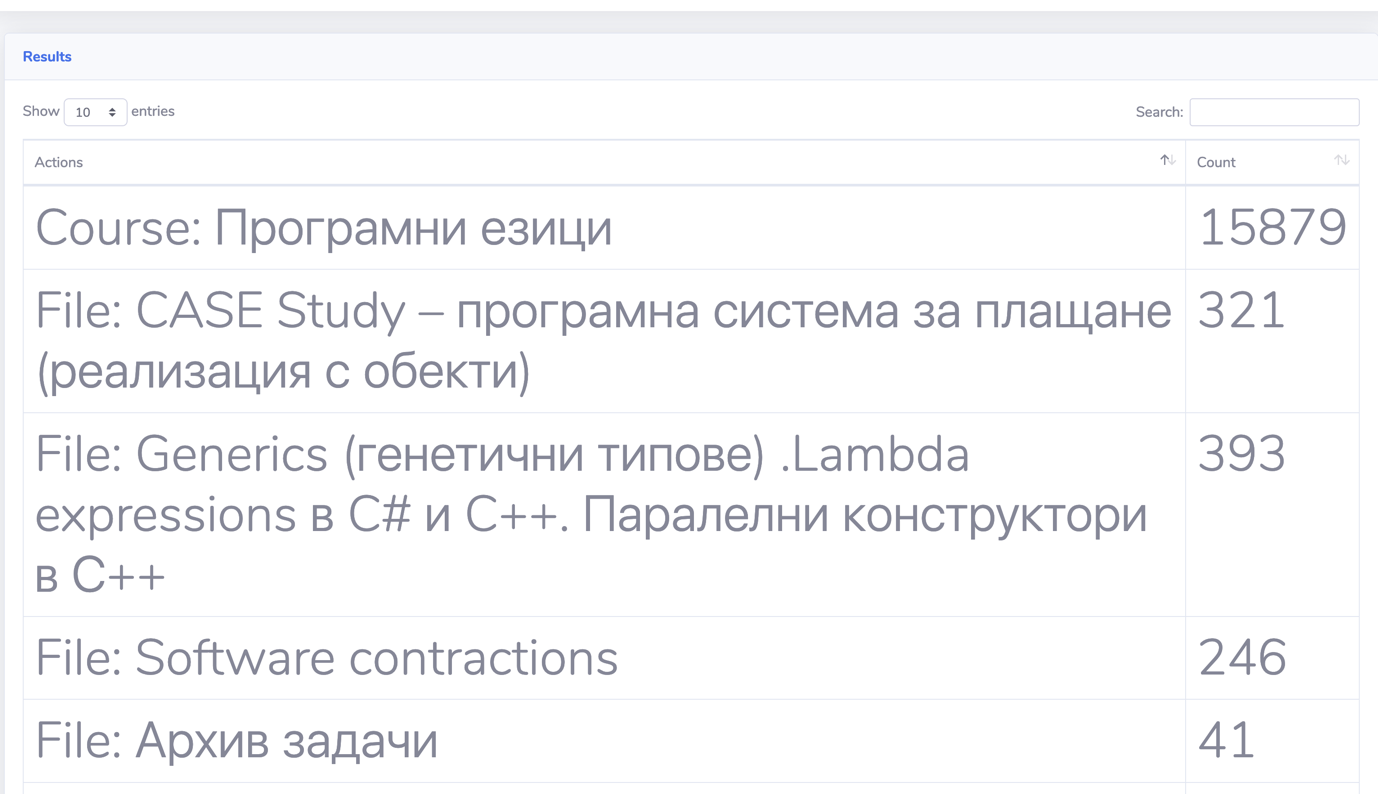


Той може да филтрира записите в таблицата по подаден текст, като напише дадения текст в “Search” полето.



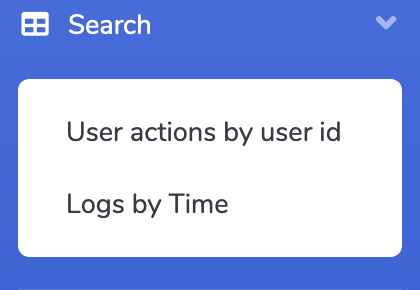


Избирайки опцията „Types of events“ потребителят бива пренасочен към страница с таблица съдържаща типовете събития и това колко често се срещат.



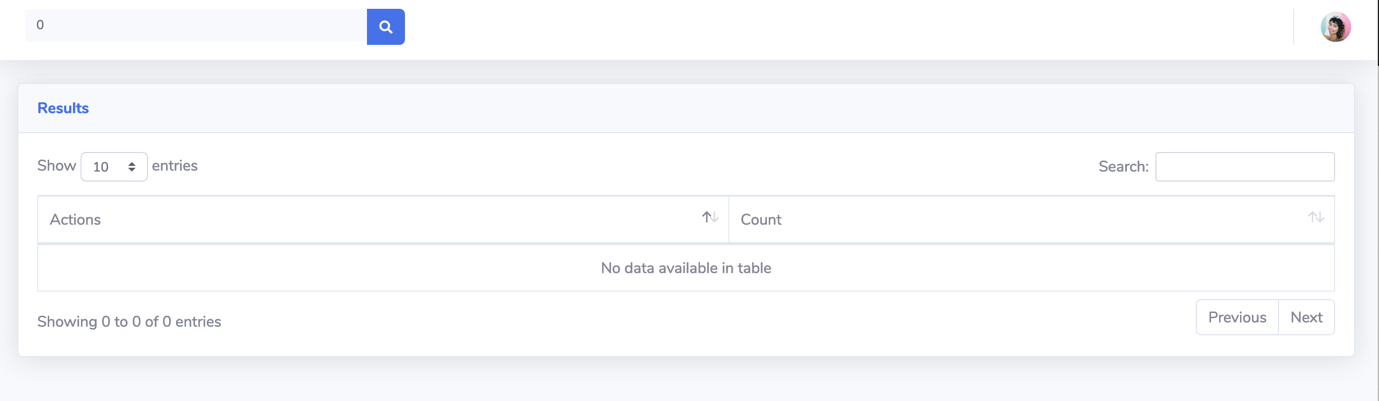
Тук той отново има гореописаните опции за търсене и това колко записан на страница да се показват.

При кликне върху текста с надпис „Search“  се появява подменю с две опции.



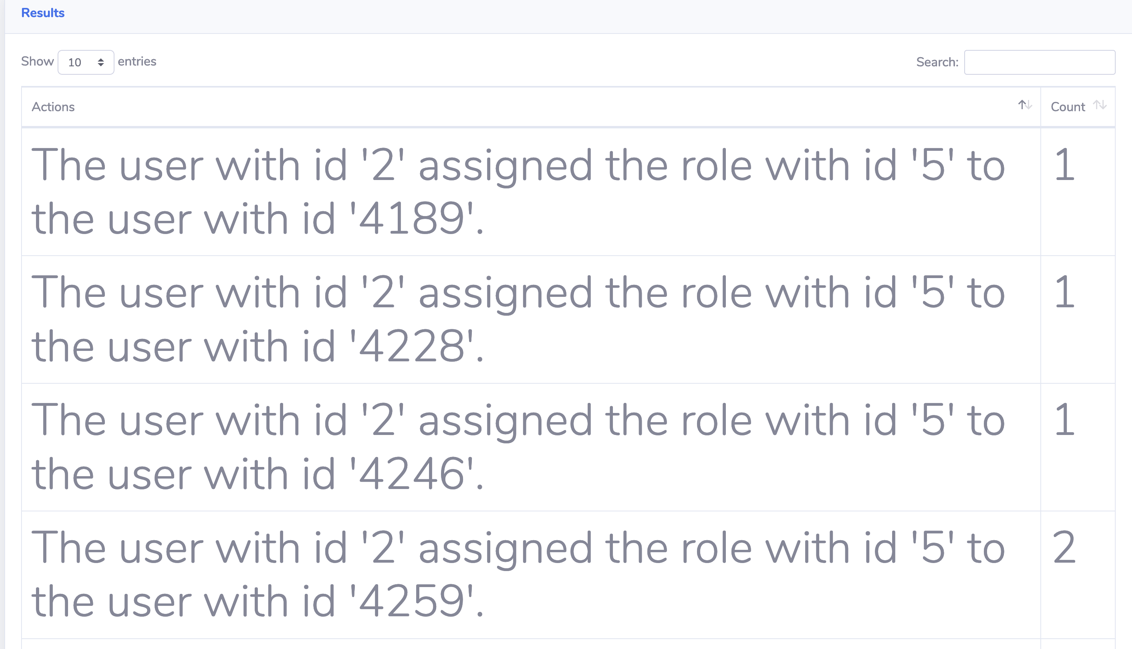
Ако потребителят избере опцията „User actions by user id“.

Той бива пренасочен към страница с празна таблица.



В полето за търсене горе в ляво се въвежда идентификационния номер на потребителя и се натиска клавишът „Enter“

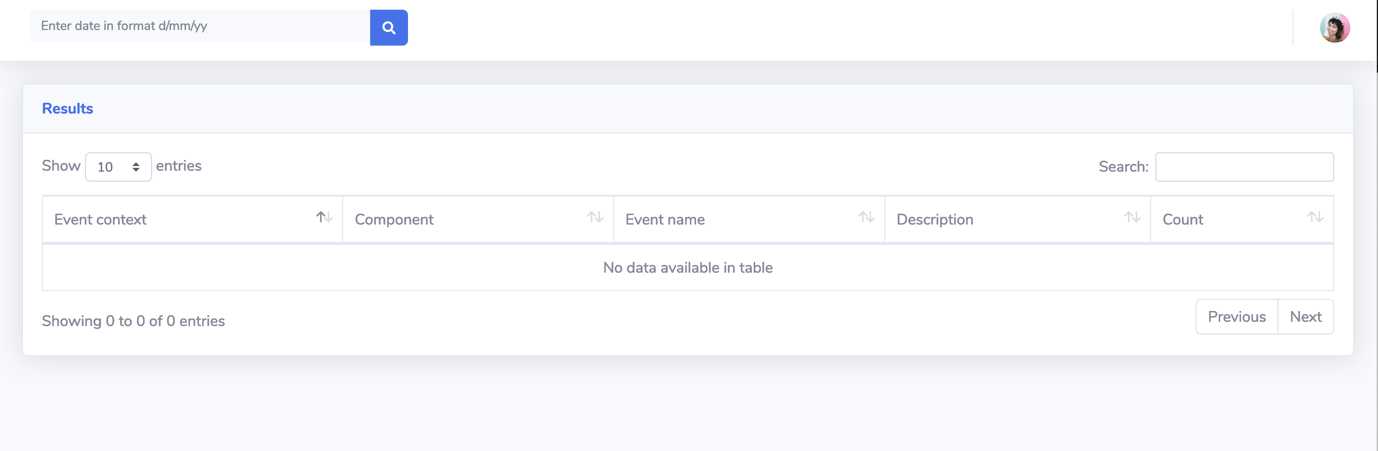
Ако бъде намерен потребител с такъв идентификационен номер таблицата се попълва с данни, ако ли не остава празна.



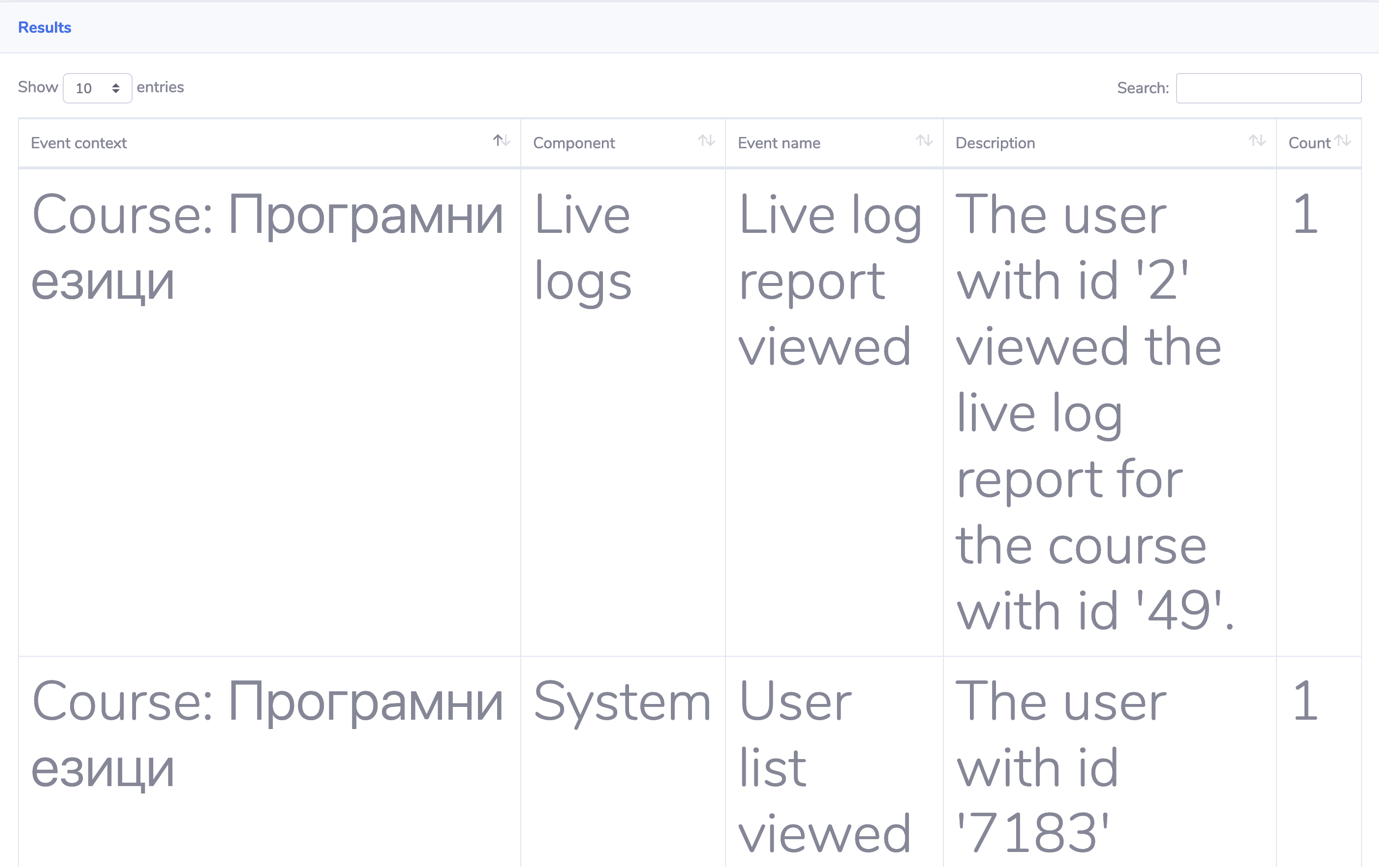
Тук той отново има гореописаните опции за търсене и това колко записан на страница да се показват.

Ако потребителят избере опцията „Logs by time“.

Той бива пренасочен към страница с празна таблица.



Горе има поле за търсене в което е описано в какъв формат трябва да е датата. Ако за тази дата бъдат намерени логове, таблицата ще се попълни, ако не ще остане празна.



И тук отново потребителят има гореописаните опции за търсене и това колко записан на страница да се показват.

* 1. Приложение

Кодът на приложението може да бъде намерен в GitHub на адрес:

<https://github.com/ElisavetaTodorova/diplomna-tu-2019>

За да се стартира приложението първо трябва или да се свали файл във „zip“ формат или да се клонира хранилището.

Свалянето на файла става като се достъпи адресът на приложението в GitHub описан по-горе.

След това се натисне зеления бутон с надпис „Clone or download“ и се избере опцията „Download ZIP“. След това съответно файлът трябва да се разархивира.

Предварителни изисквания:

1. Инсталирана минимум JAVA 8
2. Инсталиран минимум Maven 3.6.0.

Приложението може да се стартира по два начина.

1. През командния ред: след като е изтеглено от GitHub приложението трябва да се отиде в папката „diplomna-tu-2019/diplomna/restApi/“ и да се изпълни командата „mvn clean install“, а след нея и командата „mvn spring-boot:run“.
2. През „Intellij IDEA“ или друга интегрирана среда а разработка на Java, да се импортира проекта и да се стартира „main“ метода в класа „Application.java“.

Kодът на някои от основните класове.

Класът UsersActiveTime.java

**package** hadoop;  
  
**import** org.apache.hadoop.conf.Configuration;  
**import** org.apache.hadoop.fs.Path;  
**import** org.apache.hadoop.io.IntWritable;  
**import** org.apache.hadoop.io.Text;  
**import** org.apache.hadoop.mapreduce.Job;  
**import** org.apache.hadoop.mapreduce.Mapper;  
**import** org.apache.hadoop.mapreduce.Reducer;  
**import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  
**import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
  
**import** java.io.IOException;  
  
**public class** UsersActiveTime {  
  
  
 **public static class** TokenizerMapper  
 **extends** Mapper<Object, Text, Text, IntWritable> {  
  
 **private final static** IntWritable ***one*** = **new** IntWritable(1);  
 **private** Text **word** = **new** Text();  
  
 **public void** map(Object key, Text value, Context context  
 ) **throws** IOException, InterruptedException {  
 *// The user with id '7160'* String[] split = value.toString().split(**","**);  
   
 String userActiveTime = split[0];  
 context.write(**new** Text(userActiveTime), **new** IntWritable(1));  
  
 }  
  
 }  
  
 **public static class** IntSumReducer  
 **extends** Reducer<Text, IntWritable, Text, IntWritable> {  
 **private** IntWritable **result** = **new** IntWritable();  
  
 **public void** reduce(Text key, Iterable<IntWritable> values,  
 Context context  
 ) **throws** IOException, InterruptedException {  
 **int** sum = 0;  
 **for** (IntWritable val : values) {  
 sum += val.get();  
 }  
 **result**.set(sum);  
 context.write(key, **result**);  
 }  
 }  
  
  
 **public static void** main(String[] args) **throws** Exception {  
 Configuration conf = **new** Configuration();  
 Job job = Job.*getInstance*(conf, **"Users Active time"**);  
 job.setJarByClass(UsersActiveTime.**class**);  
 job.setMapperClass(UsersActiveTime.TokenizerMapper.**class**);  
 job.setCombinerClass(UsersActiveTime.IntSumReducer.**class**);  
 job.setReducerClass(UsersActiveTime.IntSumReducer.**class**);  
 job.setOutputKeyClass(Text.**class**);  
 job.setOutputValueClass(IntWritable.**class**);  
 FileInputFormat.*addInputPath*(job, **new** Path(**"/Users/i338442/diplomna/backend/src/main/resources/input"**));  
 FileOutputFormat.*setOutputPath*(job, **new** Path(**"/Users/i338442/diplomna/backend/src/main/resources/usersActiveTime"**));  
 System.*exit*(job.waitForCompletion(**true**) ? 0 : 1);  
 }  
}

MainController.java

@Controller  
@RequestMapping(value = **"/"**)  
**public class** MainController {  
  
 @Autowired  
 **private** ServiceUtil **serviceUtil**;  
  
 @GetMapping(value = **"/searchLogsGet"**)  
 **public** String searchLogsGet(@ModelAttribute(**"search"**) Search search, Model model) **throws** Exception {  
  
 **return "logsSearch"**;  
 }  
  
  
 @PostMapping(value = **"/searchLogs"**)  
 **public** String searchLogs(@ModelAttribute(**"search"**) Search search, Model model) **throws** Exception {  
  
 String data = search.getData();  
  
 List<EventLogsData> logsEvents = LogsReaderService.*getLogsForGivenDate*(data);  
 model.addAttribute(**"logsEvents"**, logsEvents);  
  
  
 **return "logsSearch"**;  
 }  
   
 @RequestMapping(value = **"/homePage"**)  
 **public** String login(Model model) **throws** IOException {  
  
 Collection<Integer> values = **serviceUtil**.getUsersActivity().values();  
 Integer[] integers = values.stream().sorted().toArray(Integer[]::**new**);  
 Map<String, Object> result = **new** HashMap<>();  
  
  
 Integer sum = values.stream()  
 .reduce(0, Integer::*sum*);  
  
  
 model.addAttribute(**"canvasData"**, **new** JSONArray(values));  
 model.addAttribute(**"mostActiveLog"**, integers[11]);  
 model.addAttribute(**"totalOfLogs"**, sum);  
 model.addAttribute(**"totalOfUsers"**, **serviceUtil**.getNumberOfUsers());  
 model.addAttribute(**"numberOfCourses"**, **serviceUtil**.getNumberOfCourses());  
 model.addAttribute(**"pieCharData"**, **new** JSONArray(**serviceUtil**.getEventsList()));  
**return "index"**;  
 }  
  
  
 @RequestMapping(value = **"/getUserActions"**, method = {RequestMethod.***GET***, RequestMethod.***POST***})  
 **public** String getUserActionsByIdMainPage(@ModelAttribute(**"search"**) Search search, Model model) **throws** Exception {  
 **return "table"**;  
 }  
  
 @PostMapping(**"/getUserActionsPost"**)  
 **public** String greetingSubmit(@ModelAttribute(**"search"**) Search search, Model model) **throws** Exception {  
  
 **int** id = search.getId();  
  
 Map<String, Integer> userActionsById = getUserActionsById(id + **""**);  
  
 List<UserAction> userActions = **new** ArrayList<>();  
  
  
 **for** (String userId : userActionsById.keySet()) {  
  
 userActions.add(**new** UserAction(userId, userActionsById.get(userId)));  
 }  
  
 model.addAttribute(**"userActions"**, userActions);  
 **return "table"**;  
 }  
  
  
 @GetMapping(value = **"/getUserIds"**)  
 **public** String getUserIds(@ModelAttribute(**"search"**) Search search, Model model) **throws** Exception {  
  
 File file = **new** File(**"/Users/i338442/diplomna/backend/src/main/resources/outputUserIds/part-r-00000"**);  
 BufferedReader br = **new** BufferedReader(**new** FileReader(file));  
  
 Map<String, Integer> result = **new** HashMap<>();  
 String st;  
 **while** ((st = br.readLine()) != **null**) {  
 String[] split = st.split(**"\t"**);  
  
 result.put(split[0], Integer.*parseInt*(split[1]));  
 }  
  
 List<UserAction> userActions = **new** ArrayList<>();  
  
 **for** (String userId : result.keySet()) {  
  
 userActions.add(**new** UserAction(userId, result.get(userId)));  
 }  
  
 model.addAttribute(**"userActions"**, userActions);  
 model.addAttribute(**"style"**, **"visibility:hidden"**);  
  
 **return "table"**;  
 }  
  
 @GetMapping(value = **"/getEvents"**)  
 **public** String getTypesOfEvents(@ModelAttribute(**"search"**) Search search, Model model) **throws** Exception {  
  
 File file = **new** File(**"/Users/i338442/diplomna/backend/src/main/resources/outputEventContext/part-r-00000"**);  
 BufferedReader br = **new** BufferedReader(**new** FileReader(file));  
  
 Map<String, Integer> result = **new** HashMap<>();  
 String st;  
 br.readLine();  
 **while** ((st = br.readLine()) != **null**) {  
 String[] split = st.split(**"\t"**);  
  
 result.put(split[0], Integer.*parseInt*(split[1]));  
 }  
  
 List<UserAction> userActions = **new** ArrayList<>();  
  
 **for** (String userId : result.keySet()) {  
  
 userActions.add(**new** UserAction(userId, result.get(userId)));  
 }  
  
 model.addAttribute(**"userActions"**, userActions);  
 model.addAttribute(**"style"**, **"visibility:hidden"**);  
  
 **return "table"**;  
 }  
  
  
  
 **private** Map<String, Integer> getUserActionsById(String userId) **throws** Exception {  
 UserWithIdActionsCount.*execute*(**new** String[]{userId});  
  
 File file = **new** File(**"/Users/i338442/diplomna/backend/src/main/resources/outputUserIdsActionsCount"** + userId + **"/part-r-00000"**);  
 BufferedReader br = **new** BufferedReader(**new** FileReader(file));  
  
 Map<String, Integer> result = **new** HashMap<>();  
 String st;  
 **while** ((st = br.readLine()) != **null**) {  
 String[] split = st.split(**"\t"**);  
  
 result.put(split[0], Integer.*parseInt*(split[1]));  
 }  
  
 **return** result;  
 }  
  
}

Моделът EventLogsData.java

**package** api.models;  
  
**public class** EventLogsData {  
  
 **private int count**;  
 **private** String **eventContext**;  
 **private** String **component**;  
 **private** String **eventName**;  
 **private** String **eventDescription**;  
  
 **public** EventLogsData(**int** count, String eventContext, String component, String eventName, String eventDescription) {  
 **this**.**count** = count;  
 **this**.**eventContext** = eventContext;  
 **this**.**component** = component;  
 **this**.**eventName** = eventName;  
 **this**.**eventDescription** = eventDescription;  
 }  
  
 **public int** getCount() {  
 **return count**;  
 }  
  
 **public void** setCount(**int** count) {  
 **this**.**count** = count;  
 }  
  
 **public** String getEventContext() {  
 **return eventContext**;  
 }  
  
 **public void** setEventContext(String eventContext) {  
 **this**.**eventContext** = eventContext;  
 }  
  
 **public** String getComponent() {  
 **return component**;  
 }  
  
 **public void** setComponent(String component) {  
 **this**.**component** = component;  
 }  
  
 **public** String getEventName() {  
 **return eventName**;  
 }  
  
 **public void** setEventName(String eventName) {  
 **this**.**eventName** = eventName;  
 }  
  
 **public** String getEventDescription() {  
 **return eventDescription**;  
 }  
  
 **public void** setEventDescription(String eventDescription) {  
 **this**.**eventDescription** = eventDescription;  
 }  
}

Сървисът LogsReaderService.java

**package** api;  
  
  
**import** api.models.EventLogsData;  
**import** hadoop.LogsPerDate;  
  
**import** java.io.BufferedReader;  
**import** java.io.File;  
**import** java.io.FileReader;  
**import** java.io.IOException;  
**import** java.util.ArrayList;  
**import** java.util.HashMap;  
**import** java.util.List;  
**import** java.util.Map;  
  
**public class** LogsReaderService {  
   
 **public static** List<EventLogsData> getLogsForGivenDate(String date) **throws** Exception {  
 File file = **new** File(**"/Users/i338442/diplomna/backend/src/main/resources/logsPerDate"** + date.replace(**"/"**, **"-"**) + **"/part-r-00000"**);  
   
 **if** (!file.exists()) {  
 LogsPerDate.*main*(**new** String[] {date});  
 }  
  
 List<EventLogsData> eventLogsDataList = **new** ArrayList<>();  
 Map<String, Integer> result = *readFromFile*(file);  
  
 **for** (String eventData: result.keySet()) {  
 **int** count = result.get(eventData);  
  
 String[] logData = eventData.split(**","**);  
  
 eventLogsDataList.add(**new** EventLogsData(count, logData[0], logData[1], logData[2], logData[3]));  
 }  
  
 **return** eventLogsDataList;  
 }  
  
 **public static void** main(String[] args) **throws** Exception {  
 LogsPerDate.*main*(**new** String[] {**"3/11/18"**});  
 }  
   
 **private static** Map<String, Integer> readFromFile(File file) **throws** IOException {  
 BufferedReader br = **new** BufferedReader(**new** FileReader(file));  
 Map<String, Integer> result = **new** HashMap<>();  
 String st;  
 br.readLine();  
 **while** ((st = br.readLine()) != **null**) {  
 String[] split = st.split(**"\t"**);  
  
 result.put(split[0], Integer.*parseInt*(split[1]));  
 }  
   
 **return** result;  
 }  
}

Класът от модулът „userFormatter“ UsersIdFormatter.java

**package** com;  
  
**import** java.io.\*;  
**import** java.util.HashMap;  
**import** java.util.LinkedHashMap;  
**import** java.util.Map;  
**import** java.util.stream.Collectors;  
  
**public class** UsersIdFormatter {  
   
 **private static** Map<Integer, Integer> *map* = **new** HashMap<Integer, Integer>();  
   
   
 **static** {  
 **try** {  
 *fillUseIdsMap*();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
   
 **public** Map<Integer, Integer> getUserIdsSortedByMostActiveUser() {  
 **final** Map<Integer, Integer> userWithMostActivity = *map*.entrySet()  
 .stream()  
 .sorted(Map.Entry.<Integer, Integer>*comparingByValue*().reversed())  
 .collect(Collectors.*toMap*(Map.Entry::getKey, Map.Entry::getValue, (e1, e2) -> e1, LinkedHashMap::**new**));  
   
 **return** userWithMostActivity;  
 }  
   
 **public** Map<Integer, Integer> getUsersIdByNameInAccedingOrder() {  
 **final** Map<Integer, Integer> userWithMostActivity1 = *map*.entrySet()  
 .stream()  
 .sorted(Map.Entry.<Integer, Integer>*comparingByKey*())  
 .collect(Collectors.*toMap*(Map.Entry::getKey, Map.Entry::getValue, (e1, e2) -> e1, LinkedHashMap::**new**));  
   
 **return** userWithMostActivity1;  
 }  
   
   
 **private static void** fillUseIdsMap() **throws** IOException {  
 File file = **new** File(**"/Users/i338442/diplomna/backend/src/main/resources/outputUserIds/part-r-00000"**);  
 BufferedReader br = **new** BufferedReader(**new** FileReader(file));  
   
  
 String st;  
 **while** ((st = br.readLine()) != **null**) {  
 String[] split = st.split(**"\\s"**);  
  
 UsersIdFormatter.*map*.put(Integer.*parseInt*(split[0]), Integer.*parseInt*(split[1]));  
 }  
 }   
}

И класът LogsPerDate.java

**package** hadoop;  
  
**import** org.apache.hadoop.conf.Configuration;  
**import** org.apache.hadoop.fs.Path;  
**import** org.apache.hadoop.io.IntWritable;  
**import** org.apache.hadoop.io.Text;  
**import** org.apache.hadoop.mapreduce.Job;  
**import** org.apache.hadoop.mapreduce.Mapper;  
**import** org.apache.hadoop.mapreduce.Reducer;  
**import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  
**import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
  
**import** java.io.IOException;  
  
**public class** LogsPerDate {  
  
 **private static** String *date*;  
  
 **public static class** TokenizerMapper  
 **extends** Mapper<Object, Text, Text, IntWritable> {  
  
 **private final static** IntWritable ***one*** = **new** IntWritable(1);  
 **private** Text **word** = **new** Text();  
  
 **public void** map(Object key, Text value, Context context  
 ) **throws** IOException, InterruptedException {  
  
  
 **if** (value.toString().contains(*date*)) {  
 String[] split = value.toString().split(**","**);  
  
 context.write(**new** Text(String.*format*(**"%s,%s,%s,%s,%s"**, split[2], split[3], split[4], split[5], split[6])), **new** IntWritable(1));  
 }  
  
 }  
  
 }  
  
 **public static class** IntSumReducer  
 **extends** Reducer<Text, IntWritable, Text, IntWritable> {  
 **private** IntWritable **result** = **new** IntWritable();  
  
 **public void** reduce(Text key, Iterable<IntWritable> values,  
 Context context  
 ) **throws** IOException, InterruptedException {  
 **int** sum = 0;  
 **for** (IntWritable val : values) {  
 sum += val.get();  
 }  
 **result**.set(sum);  
 context.write(key, **result**);  
 }  
 }  
  
  
 **public static void** main(String[] args) **throws** Exception {  
 *date* = args[0];  
 Configuration conf = **new** Configuration();  
 Job job = Job.*getInstance*(conf, **"LogsPreDate"** + *date*);  
 job.setJarByClass(LogsPerDate.**class**);  
 job.setMapperClass(LogsPerDate.TokenizerMapper.**class**);  
 job.setCombinerClass(LogsPerDate.IntSumReducer.**class**);  
 job.setReducerClass(LogsPerDate.IntSumReducer.**class**);  
 job.setOutputKeyClass(Text.**class**);  
 job.setOutputValueClass(IntWritable.**class**);  
 FileInputFormat.*addInputPath*(job, **new** Path(**"/Users/i338442/diplomna/backend/src/main/resources/input"**));  
 FileOutputFormat.*setOutputPath*(job, **new** Path(**"/Users/i338442/diplomna/backend/src/main/resources/logsPerDate"** + (*date*.replace(**"/"**, **"-"**))));  
  
 job.waitForCompletion(**true**);  
 }  
}

table.html

<!DOCTYPE **html**>  
<**html lang="en" xmlns:th="http://www.thymeleaf.org"**>  
  
<**head**>  
  
 <**meta charset="utf-8"**>  
 <**meta http-equiv="X-UA-Compatible" content="IE=edge"**>  
 <**meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no"**>  
 <**meta name="description" content=""**>  
 <**meta name="author" content=""**>  
  
 <**title**>SB Admin 2 - Tables</**title**>  
  
 *<!-- Custom fonts for this template -->* <**link th:href="@{/vendor/fontawesome-free/css/all.min.css}" rel="stylesheet" type="text/css"**>  
 <**link href="https://fonts.googleapis.com/css?family=Nunito:200,200i,300,300i,400,400i,600,600i,700,700i,800,800i,900,900i" rel="stylesheet"**>  
  
 *<!-- Custom styles for this template -->* <**link th:href="@{/css/sb-admin-2.min.css}" rel="stylesheet"**>  
  
 *<!-- Custom styles for this page -->* <**link th:href="@{/vendor/datatables/dataTables.bootstrap4.min.css}" rel="stylesheet"**>  
  
</**head**>  
  
<**body id="page-top"**>  
  
*<!-- Page Wrapper -->*<**div id="wrapper"**>  
  
 *<!-- Sidebar -->* <**ul class="navbar-nav bg-gradient-primary sidebar sidebar-dark accordion" id="accordionSidebar"**>  
  
 *<!-- Sidebar - Brand -->* <**a class="sidebar-brand d-flex align-items-center justify-content-center" href="/homePage"**>  
 <**div class="sidebar-brand-icon rotate-n-15"**>  
 <**i class="fas fa-laugh-wink"**></**i**>  
 </**div**>  
 <**div class="sidebar-brand-text mx-3"**>SB Admin <**sup**>2</**sup**></**div**>  
 </**a**>  
  
 *<!-- Divider -->* <**hr class="sidebar-divider my-0"**>  
  
 *<!-- Nav Item - Dashboard -->* <**li class="nav-item"**>  
 <**a class="nav-link" href="/homePage"**>  
 <**i class="fas fa-fw fa-tachometer-alt"**></**i**>  
 <**span**>Dashboard</**span**></**a**>  
 </**li**>  
  
 *<!-- Divider -->* <**hr class="sidebar-divider"**>  
  
 *<!-- Heading -->* <**div class="sidebar-heading"**>  
 Interface  
 </**div**>  
  
 *<!-- Nav Item - Pages Collapse Menu -->* <**li class="nav-item"**>  
 <**a class="nav-link collapsed" href="#" data-toggle="collapse" data-target="#collapseTwo" aria-expanded="true" aria-controls="collapseTwo"**>  
 <**i class="fas fa-fw fa-table"**></**i**>  
 <**span**>View</**span**>  
 </**a**>  
 <**div id="collapseTwo" class="collapse" aria-labelledby="headingTwo" data-parent="#accordionSidebar"**>  
 <**div class="bg-white py-2 collapse-inner rounded"**>  
 <**h6 class="collapse-header"**>Custom Components:</**h6**>  
 <**a class="collapse-item" href="/getUserIds"**>All Users Ids</**a**>  
 <**a class="collapse-item" href="/getEvents"**>Types of events</**a**>  
 </**div**>  
 </**div**>  
 </**li**>  
  
 *<!-- Nav Item - Utilities Collapse Menu -->* <**li class="nav-item"**>  
 <**a class="nav-link collapsed" href="#" data-toggle="collapse" data-target="#collapseUtilities" aria-expanded="true" aria-controls="collapseUtilities"**>  
 <**i class="fas fa-fw fa-table"**></**i**>  
 <**span**>Search</**span**>  
 </**a**>  
 <**div id="collapseUtilities" class="collapse" aria-labelledby="headingUtilities" data-parent="#accordionSidebar"**>  
 <**div class="bg-white py-2 collapse-inner rounded"**>  
 *<!--<h6 class="collapse-header">Custom Utilities:</h6>-->* <**a class="collapse-item" href="/getUserActions"**>User actions by user id</**a**>  
 <**a class="collapse-item" href="/searchLogsGet"**>Logs by Time</**a**>  
 </**div**>  
 </**div**>  
 </**li**>  
  
 *<!-- Divider -->* <**hr class="sidebar-divider"**>  
  
 *<!-- Sidebar Toggler (Sidebar) -->* <**div class="text-center d-none d-md-inline"**>  
 <**button class="rounded-circle border-0" id="sidebarToggle"**></**button**>  
 </**div**>  
  
 </**ul**>  
 *<!-- End of Sidebar -->  
  
 <!-- Content Wrapper -->* <**div id="content-wrapper" class="d-flex flex-column"**>  
  
 *<!-- Main Content -->* <**div id="content"**>  
  
 *<!-- Topbar -->* <**nav class="navbar navbar-expand navbar-light bg-white topbar mb-4 static-top shadow"**>  
  
 *<!-- Sidebar Toggle (Topbar) -->* <**button id="sidebarToggleTop" class="btn btn-link d-md-none rounded-circle mr-3"**>  
 <**i class="fa fa-bars"**></**i**>  
 </**button**>  
  
 *<!--th:style="visibility:${visibility}">-->  
 <!-- Topbar Search -->* <**form class="d-none d-sm-inline-block form-inline mr-auto ml-md-3 my-2 my-md-0 mw-100 navbar-search" action="#" th:action="@{/getUserActionsPost}" th:object="${search}" method="post" id="searchFormId" th:style="${style}"**>  
 <**div class="input-group"**>  
 <**input type="text" class="form-control bg-light border-0 small" placeholder="Search for..." aria-label="Search" aria-describedby="basic-addon2" th:field="\*{id}"**>  
 <**div class="input-group-append"**>  
 <**button class="btn btn-primary" type="button"**>  
 <**i class="fas fa-search fa-sm"**></**i**>  
 </**button**>  
 </**div**>  
 </**div**>  
 </**form**>  
   
 *<!-- Topbar Navbar -->* <**ul class="navbar-nav ml-auto"**>  
  
 *<!-- Nav Item - Search Dropdown (Visible Only XS) -->* <**li class="nav-item dropdown no-arrow d-sm-none"**>  
 <**a class="nav-link dropdown-toggle" href="#" id="searchDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false"**>  
 <**i class="fas fa-search fa-fw"**></**i**>  
 </**a**>  
 *<!-- Dropdown - Messages -->* <**div class="dropdown-menu dropdown-menu-right p-3 shadow animated--grow-in" aria-labelledby="searchDropdown"**>  
 <**form class="form-inline mr-auto w-100 navbar-search"**>  
 <**div class="input-group"**>  
 <**input type="text" class="form-control bg-light border-0 small" placeholder="Search for..." aria-label="Search" aria-describedby="basic-addon2"**>  
 <**div class="input-group-append"**>  
 <**button class="btn btn-primary" type="button"**>  
 <**i class="fas fa-search fa-sm"**></**i**>  
 </**button**>  
 </**div**>  
 </**div**>  
 </**form**>  
 </**div**>  
 </**li**>  
   
  
 <**div class="topbar-divider d-none d-sm-block"**></**div**>  
  
 *<!-- Nav Item - User Information -->* <**li class="nav-item dropdown no-arrow"**>  
 <**a class="nav-link dropdown-toggle" href="#" id="userDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false"**>  
 *<!--<span class="mr-2 d-none d-lg-inline text-gray-600 small">Valerie Luna</span>-->* <**img class="img-profile rounded-circle" src="https://source.unsplash.com/QAB-WJcbgJk/60x60"**>  
 </**a**>  
 *<!-- Dropdown - User Information -->* <**div class="dropdown-menu dropdown-menu-right shadow animated--grow-in" aria-labelledby="userDropdown"**>  
   
 <**a class="dropdown-item" href="#" data-toggle="modal" data-target="#logoutModal"**>  
 <**i class="fas fa-sign-out-alt fa-sm fa-fw mr-2 text-gray-400"**></**i**>  
 Logout  
 </**a**>  
 </**div**>  
 </**li**>  
  
 </**ul**>  
 </**nav**>  
 *<!-- End of Topbar -->  
 <!-- Begin Page Content -->* <**div class="container-fluid"**>  
 *<!-- DataTales Example -->* <**div class="card shadow mb-4"**>  
 <**div class="card-header py-3"**>  
 <**h6 class="m-0 font-weight-bold text-primary"**>Results</**h6**>  
 </**div**>  
 <**div class="card-body"**>  
 <**div class="table-responsive"**>  
 <**table class="table table-bordered" id="dataTable" width="100%" cellspacing="0"**>  
 <**thead**>  
 <**tr**>  
 <**td**>Actions</**td**>  
 <**td**>Count</**td**>  
 </**tr**>   
 </**thead**>  
 <**tbody**>  
 <**tr th:each="userAction: ${userActions}"**>  
 <**td class="display-4" th:text="${userAction.actionName}"**>UserId</**td**>  
 <**td class="display-4" th:text="${userAction.count}"**>UserId</**td**>  
  
 </**tr**>  
 </**tbody**>  
  
 </**table**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
  
 </**div**>  
 *<!-- End of Main Content -->  
  
 <!-- Footer -->* <**footer class="sticky-footer bg-white"**>  
 <**div class="container my-auto"**>  
 <**div class="copyright text-center my-auto"**>  
 <**span**>Copyright **&copy;** Your Website 2019</**span**>  
 </**div**>  
 </**div**>  
 </**footer**>  
 *<!-- End of Footer -->* </**div**>  
 *<!-- End of Content Wrapper -->*</**div**>  
*<!-- End of Page Wrapper -->  
<!-- Scroll to Top Button-->*<**a class="scroll-to-top rounded" href="#page-top"**>  
 <**i class="fas fa-angle-up"**></**i**>  
</**a**>  
*<!-- Logout Modal-->*<**div class="modal fade" id="logoutModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true"**>  
 <**div class="modal-dialog" role="document"**>  
 <**div class="modal-content"**>  
 <**div class="modal-header"**>  
 <**h5 class="modal-title" id="exampleModalLabel"**>Ready to Leave?</**h5**>  
 <**button class="close" type="button" data-dismiss="modal" aria-label="Close"**>  
 <**span aria-hidden="true"**>×</**span**>  
 </**button**>  
 </**div**>  
 <**div class="modal-body"**>Select "Logout" below if you are ready to end your current session.</**div**>  
 <**div class="modal-footer"**>  
 <**button class="btn btn-secondary" type="button" data-dismiss="modal"**>Cancel</**button**>  
 <**a class="btn btn-primary" href="https://elisaveta.accounts400.ondemand.com/service/me/logout"**>Logout</**a**>  
 </**div**>  
 </**div**>  
 </**div**>  
</**div**>  
*<!-- Bootstrap core JavaScript-->*<**script th:src="@{/vendor/jquery/jquery.min.js}"**></**script**>  
<**script th:src="@{/vendor/bootstrap/js/bootstrap.bundle.min.js}"**></**script**>  
*<!-- Core plugin JavaScript-->*<**script th:src="@{/vendor/jquery-easing/jquery.easing.min.js}"**></**script**>  
*<!-- Custom scripts for all pages-->*<**script th:src="@{/js/sb-admin-2.min.js}"**></**script**>  
*<!-- Page level plugins -->*<**script th:src="@{/vendor/datatables/jquery.dataTables.min.js}"**></**script**>  
<**script th:src="@{/vendor/datatables/dataTables.bootstrap4.min.js}"**></**script**>  
*<!-- Page level custom scripts -->*<**script th:src="@{/js/demo/datatables-demo.js}"**></**script**>  
</**body**>  
  
</**html**>

Кодът на останалите java класове, както и всички файлове служещи за имплементацията на уеб интерфейса могат да бъдат намерени в посочения адрес в GitHub.

* 1. Използвани източници:

1. EMC https://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf
2. IBM <https://public.dhe.ibm.com/common/ssi/ecm/wr/en/wrl12345usen/watson-customer-engagement-watson-marketing-wr-other-papers-and-reports-wrl12345usen-20170719.pdf>
3. <http://www.physics.org/thankphysics/internet/>
4. Data never sleeps 5.0 <https://www.domo.com/learn/data-never-sleeps-5?aid=ogsm072517_1&sf100871281=1>
5. The Guardian https://www.theguardian.com/news/datablog/2012/dec/19/big-data-study-digital-universe-global-volume
6. Global Web Index <https://blog.globalwebindex.com/chart-of-the-day/social-media-captures-30-of-online-time/>
7. Inside big data <https://blog.globalwebindex.com/chart-of-the-day/social-media-captures-30-of-online-time/>
8. <https://en.wikipedia.org/wiki/Apache_Hadoop>
9. <https://en.wikipedia.org/wiki/Spring_Framework>
10. <https://www.guru99.com/introduction-to-mapreduce.html>
11. <https://en.wikipedia.org/wiki/Thymeleaf>
12. <https://help.sap.com/viewer/6d6d63354d1242d185ab4830fc04feb1/Cloud/en-US>
13. Java development kit 8 <https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>
14. Maven version 3.6.0 <https://maven.apache.org/docs/3.6.0/release-notes.html>
15. Intellij IDEA <https://www.jetbrains.com/idea/download/>
16. Spring-Boot documentation <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>
17. Apache Hadoop Documentation <https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html>