МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт Математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки / специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль) /специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

на тему «Системное моделирование поиска событий в базе СКУД »

Обучающийся А.С. Овчинникова _____
(Инициалы Фамилия) (личная подпись)

Руководитель канд.пед.наук, доцент, Т.А. Агошкова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант канд.пед.наук, доцент, Т.С. Якушева
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2022

Аннотация

В работе исследуются характеристики математической модели системы информационного поиска Web-ресурсов.

Выпускная квалификационная работа состоит из введения, четырех глав, заключения, приложений. Общий объем работы 45 страниц, 29 рисунков, 2 раздела, 28 использованных источников.

Цель работы – создание Web-приложения для Тольяттинского государственного университета на языке PHP, в котором будет осуществляться поиск в базе данных системы контроля и управления доступом.

Объект исследования – события в базе системы контроля и управления доступом.

Предмет исследования – Web-приложение поиска в базе системы контроля и управления доступом.

В результате исследования решены следующие задачи: спроектировано и разработано Web-приложения для поиска событий в базе системы контроля и управления доступом; произведено тестирование созданного приложения.

Значимость работы – позволяет сотрудникам Тольяттинского государственного университета осуществлять удобный поиск по событиям удаленно.

Annotation

The paper studies the characteristics of the mathematical model of the information retrieval system for Web-resources.

The final qualifying work consists of an introduction, four chapters, a conclusion, and applications. The total volume of work is 45 pages, 29 figures, 2 sections, 28 references.

The purpose of the work is to create a Web-application for Togliatti State University in PHP, which will search the database of the access control and management system.

The object of research is the events in the base of the access control and management system.

The subject of the study is a Web-application for searching in the database of an access control and management system.

As a result of the study, the following tasks were solved: Web-applications were designed and developed to search for events in the database of the access control and management system; tested the created application.

The significance of the work - allows employees of Togliatti State University to carry out a convenient search for events remotely.

Содержание

Введение

В настоящее время ТГУ использует систему контроля и управления доступом APACS 3000. Система выполняет несколько функций: учет рабочего времени, контроль доступа, контроль времянахождения на объекте. Ее интерфейс выполнен в стиле стандартных приложений для работы на Microsoft Windows. Невозможность использования системы для поиска событий на сторонних операционных системах, а также иных устройств, таких как планшет, смартфон является недостатком. Рассматривается именно поиск по событиям, так как более расширенное Web-приложение с управлением процесса доступа и настройками оборудования, то есть полным интерфейсом, небезопасно из-за возможных кибер-атак или получения доступа сторонним лицам.

Разработка Web-приложения позволит получать доступ к событиям из любого места в сети. Важным преимуществом такого Web-приложения является независимость от операционной системы. Web-приложения для поиска событий в базе системы контроля и управлением доступом будет выводить события по таким запросам: дата и ФИО.

Объектом исследования являются события в базе системы контроля и управления доступом.

Предметом исследования является алгоритмы поиска в базе системы контроля и управления доступом.

Целью выпускной квалификационной работы является построение модели поиска событий в базе СКУД и апробация ее результативности через веб-приложение.

Для достижения цели выпускной квалификационной работы необходимо решить следующие задачи:

- выбрать учебную и учебно-методическую литературы по системному моделированию;
- выполнить концептуальное моделирование предметной области;

- проанализировать существующие разработки и обоснование выбора технологии проектирования;
- написать техническое задание;
- разработать математические модели поиска;
- обосновать средства реализации математическо-аналитической системы;
- определить архитектуру математическо-аналитической системы;
- разработать Web-приложение поиска событий в базе СКУД;
- протестировать работоспособность Web-приложения;

В выпускной квалификационной работе рассматриваются вопросы по разработке и реализации системного моделирования поиска событий в базе СКУД.

В заключении представлены результаты и выводы о выполненной работе.

В первом разделе проведено моделирование трех математических систем поиска.

Во втором разделе описаны средства разработки, используемые для создания Web-приложения, разработка и реализация Web-приложения.

В третьем разделе произведено тестирование Web-приложения, показана его результативность.

Итогом выпускной квалификационной работы является выполнение формального моделирование предметной области, реализация Web-приложения для поиска событий в базе системы контроля и управления доступом.

1 Моделирование поиска событий

1.1 Характеристика объекта исследования

В настоящее время почти все приложения, обрабатывающие информацию в Интернете, были бы абсолютно несостоятельны без поддержки технологии поиска информации. Для нахождения информации необходимы поисковые системы. Рассматривать теорию поиска информации нужно для того, чтобы не появлялись трудности для информационного поиска. Ведь объемы информации, в которых производится поиск, постоянно увеличивается, поэтому нужно скрупулёзное эмпирическое тестирование и эксперименты, подкрепленные теорией поиска информации.

В данном разделе описаны и разъяснены три ключевые модели поиска. Для понимания поиска информации необходимо узнать об моделях поиска, которые отличаются друг от друга. Из-за различности моделей для разработки определенных методов поиска информации подходят одни модели, но для иных методов подходят уже другие модели.

Модели поиска информации моделируют информационно-поисковую систему. Она представляет собой программное обеспечение, задачей которого является управление и хранение документов, а также помощь для пользователя в поиске необходимой ему информации. Документы могут обычно текстовые, иногда бывают мультимедийные. Однако информационно-поисковая система не дает ответ на вопросы и неявно возвращает информацию, а сообщает о наличии и расположении документов, которые, возможно, содержат необходимую информацию. Если документ будет соответствовать запросу пользователя в необходимой информации, то он будет называться соответствующим. Совершенная информационно-поисковая система должна получать исключительно релевантные документы без лишних документов.

Два пользователя, которые зададут одинаковый запрос в информационно-поисковую систему, могут по-разному определить уместность полученных документов. Получается, что оценка полученного результата является субъективной. Ведь для одного результат покажется релевантным, а для другого неправильным. Вследствие чего, очевидно, что совершенных информационно-поисковых систем не может быть и не будет.

Информационно-поисковая система должна поддерживать три основных процесса: представление содержимого документов, представление информационных потребностей пользователя и сравнение двух представлений [4].

Процессы визуализированы на рисунке 1. На рисунке 1 квадратные прямоугольники представляют данные, а закругленные прямоугольники представляют процессы.

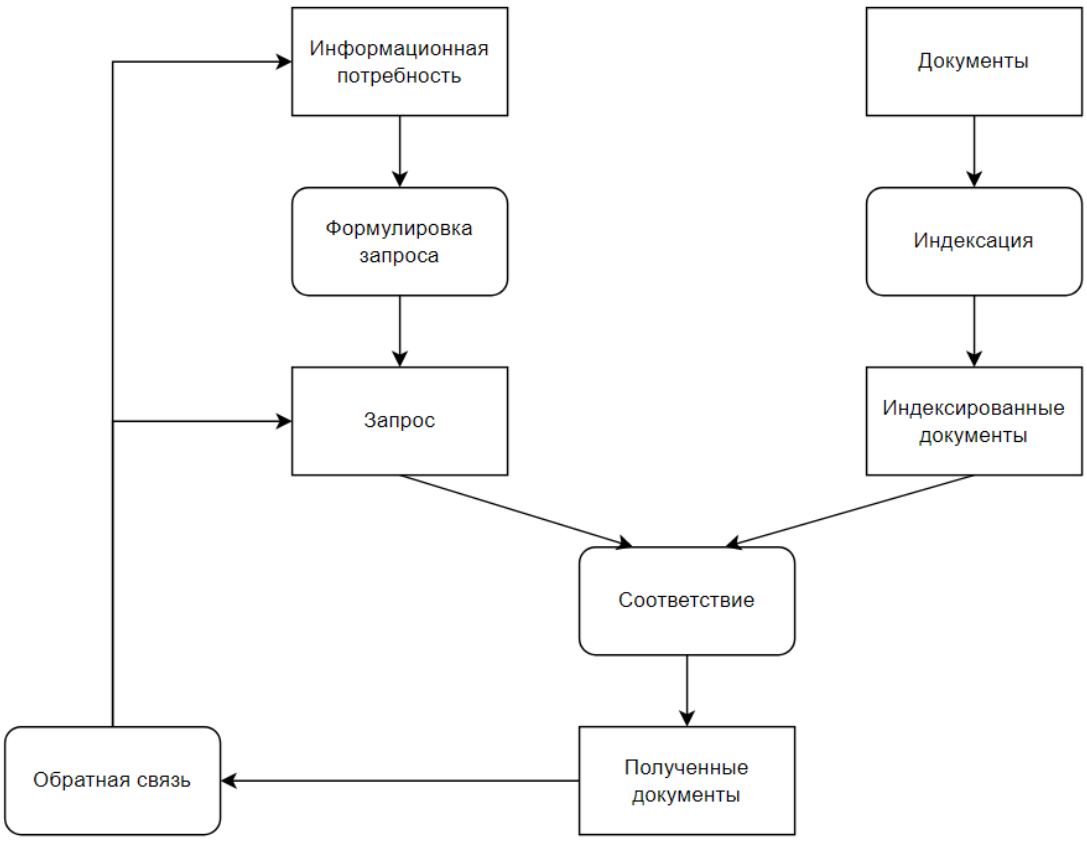


Рисунок 1 - Процессы поиска информации

Представление документов обычно называется процессом индексации. Процесс происходит в автономном режиме, то есть конечный пользователь информационно-поисковой системы не участвует напрямую. Результатом процесса индексации является представление документа. Часто полнотекстовые поисковые системы используют довольно тривиальный алгоритм для получения индексных представлений, например алгоритм, который идентифицирует слова в тексте и переводит их в нижний регистр. Процесс индексации может включать фактическое хранение документа в системе, но часто документы сохраняется только частично, например, только название и аннотация, плюс информация о фактическом местоположении документа.

Пользователи ищут не просто для развлечения, у них есть потребность в информации. Процесс представления их информационных потребностей часто называют процессом формулирования запроса. Результирующее представление — это запрос. В широком смысле формулировка запроса может означать полный интерактивный диалог между системой и пользователем, ведущий не только к подходящему запросу, но, возможно, также к лучшему пониманию пользователем его/ее потребности в информации: это обозначается процесс обратной связи на рисунке 1.

Сравнение запроса с представлениями документа называется процессом сопоставления. Процесс сопоставления обычно приводит к ранжированному списку документов. Пользователи будут перемещаться по этому списку документов в поисках необходимой им информации. Ранжированный поиск приведет к тому, что соответствующие документы окажутся в верхней части ранжированного списка, что сведет к минимуму время, которое пользователь должен потратить на чтение документов. Алгоритмы ранжирования, основанные на статистических подходах, легко сокращают вдвое время, которое пользователь тратит на чтение документов. Теория, лежащая в основе алгоритмов ранжирования, является важнейшей частью поиска информации и основной темой этого раздела.

Причина для создания моделей поиска информации – модели могут служить основой для реализации реальной поисковой системы.

Джурд Химста писал: «Математические модели используются во многих научных областях с целью понимания и обоснования некоторого поведения или явления в реальном мире. Модель поиска информации предсказывает и объясняет, что пользователь сочтет релевантным с учетом пользовательского запроса. Правильность предсказаний модели может быть проверена в контролируемом эксперименте. Чтобы делать прогнозы и лучше понимать поиск информации, модели должны быть прочно основаны на интуиции, метафорах и некоторых разделах математики. Интуиция важна, потому что она помогает получить модель, признанную исследовательским сообществом разумной. Метафоры важны, потому что они помогают объяснить значение модели более широкой аудитории. Математика необходима для формализации модели, обеспечения согласованности и обеспечения того, чтобы она могла быть реализована в реальной системе. Как таковая, модель поиска информации служит основой, которая используется для реализации реальной системы поиска информации» [20].

В следующих разделах будет довольно подробно описано в общей сложности три модели поиска информации. В литературе по поиску информации было предложено еще много моделей, но выбор, сделанный в этом разделе, рассматривает три ключевых типа подходов к моделированию.

1.2 Моделирование системы поиска

1.2.1 Булева модель точного соответствия

В данном разделе рассмотрим модель информационного поиска, которая гарантирует точное соответствие, т.е. документы либо извлекаются, либо нет, но извлеченные документы не ранжируются.

Булева модель — это первая модель информационного поиска. Модель может быть объяснена, если рассматривать термин запроса как однозначное

определение набора документов. Например, запрос термин Имя формирует набор всех документов, которые индексируются с термином Имя. Применяя операторы математической логики Джорджа Буля, термины запроса и надлежащие им наборы документов могут быть совмещены, чтобы формировать новые наборы документов. Буль определил три основных оператора: логическое произведение, называемое И, обозначаемое как ∧, логическая сумма, называемая ИЛИ, обозначаемая как ∨, и логическая разность, называемая НЕ, обозначаемая как ¬. Объединение терминов с помощью оператора И определяет набор документов, который меньше или равен набору документов любого из отдельных терминов. Например, запрос Фамилия И Имя получит набор документов, которые индексируются как термином Фамилия, так и термином Имя, т.е. пересечение обоих наборов. Объединение терминов с помощью оператора ИЛИ определяет набор документов, который больше или равен набору документов любого из отдельных терминов.

Таким образом, запрос Имя ИЛИ Отчество определит набор документов, которые индексируются либо одним из этих терминов, либо одним из двух. документов, которые индексируются либо термином Имя, либо термином Отчество, или и тем, и другим, т.е. объединение обоих наборов. Это наглядно представлено на диаграммах Эйлера-Венна на рисунках 2, 3, 4, на котором каждый набор документов изображен в виде круга. Пересечения этих кругов делят коллекцию документов на восьми непересекающихся областей, объединение которых дает двести пятьдесят шесть различных булевых комбинаций Фамилия, Имя и Отчество.

На рисунках 2, 3 и 4 найденные наборы визуализированы закрашенными областями.

На рисунке 2 показана визуализация логического произведения.

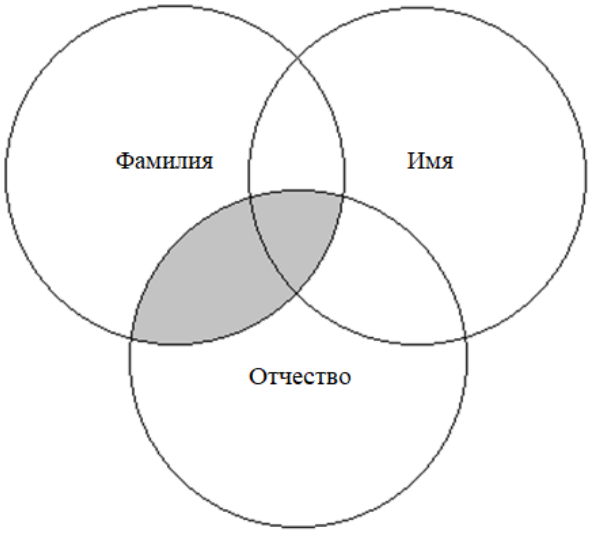


Рисунок 2 – Диаграмма Эйлера-Венна, которая отображает выражение *Фамилия И Отчество (Фамилия ∧ Отчество),* то есть логическое произведение

Далее описание второй диаграммы Эйлера-Венна (рисунок 3), которая отображает логическую сумму.

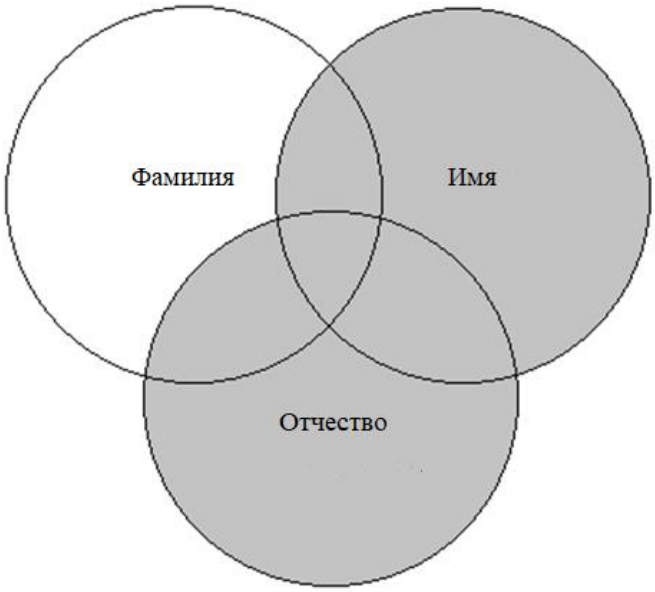


Рисунок 3 – Диаграмма Эйлера-Венна, которая отображает выражение *Имя ИЛИ Отчество (ИМЯ ∨ Отчество),* то есть логическую сумму

Далее описание диаграммы Эйлера-Венна (рисунок 4), которая отображает логическое произведение, а также логическую разность.

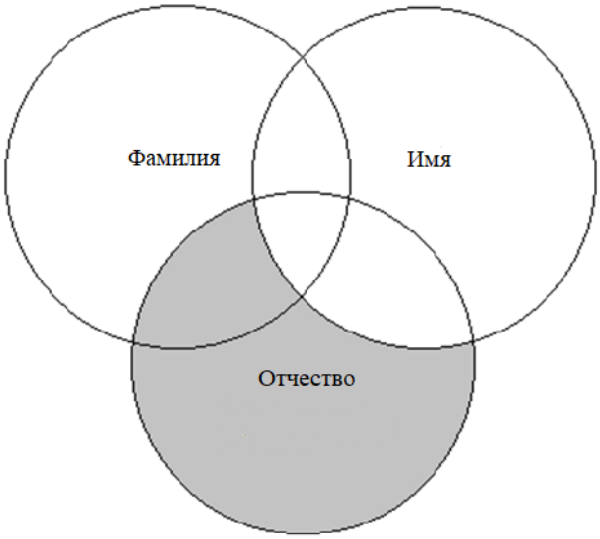


Рисунок 4 – Диаграмма Эйлера-Венна, которая отображает выражение *Отчество ИЛИ НЕ Имя (Отчество ∨ ¬Имя)*, то есть логическое произведение, а также логическую разность Имя

Достоинством булевой модели считается то, что она дает пользователям ощущение контроля над системой. Незамедлительно становится понятно, из-за чего документ был получен в результате запроса. В случае если набор документов либо слишком мал, либо слишком великий, то незамедлительно ясно, какие операторы дадут соответственно больший или малый набор. Главный минус модели это – отсутствие ранжировки соответствующих документов, потому что модель или извлекает документ, или нет. Из-за этого система может выявлять неправильные наборы документов.

1.2.2 Векторная модель

Следуя критерию подобия, Луна, первым шагом является подсчет количества элементов, которые разделяют запрос и индексное представление документа. Если индексное представление документа представляет собой вектор $\vec{d} = (d_1, d_2, \dots, d_m)$ каждый компонент которого d_k ($1 \leq k \leq m$) связан с

термином индекса; и если запрос представляет собой аналогичный вектор $\vec{q} = (q_1, q_2 q)$, компоненты которого связаны с одними и теми же терминами, тогда прямой мерой сходства является векторное внутреннее произведение (1):

$$\text{score}(\vec{d}, \vec{q}) = \sum_{k=1}^{m} d_k * q_k \quad (1)$$

где, $\vec{d}$ – вектор, представляющий собой индексное представление документа,

d_k - компоненты индексного представления документа,

q_k - компоненты индексного представления запроса.

Если вектор содержит двоичные компоненты, т.е. значение компонента равно 1, если термин встречается в документе или запросе, и 0, если нет, то векторное произведение измеряет количество общих терминов. В более общем представлении для компонентов векторов $\vec{d}$ и $\vec{q}$ использовались бы натуральные числа или действительные числа.

На основе критерия подобия Питера Луна была предложена модель Джерардом Солтоном и Майклом МакГиллом в 1983 году. Они рассматривали документ и запрос как векторы, встроенные в многомерное евклидово пространство, где каждому члену присваивается отдельное измерение. Сходство между документом и запросом задавалось косинусом между вектором термина и вектором. Косинусное сходство — это мера сходства между двумя векторами, которая применяется для измерения косинуса угла между ними. Если даны два вектора признаков d и q, то косинусное сходство, cos(0), может быть представлено используя скалярное произведение и норму по формуле (2):

$$\frac{\sum_{k=1}^{m} d_k * q_k}{\sqrt{\sum_{k=1}^{m} (d_k)^2} * \sqrt{\sum_{k=1}^{m} (q_k)^2}} \quad (2)$$

где d_k - компоненты индексного представления документа,

q_k - компоненты индексного представления запроса.

Метафора углов между векторами в многомерном пространстве позволяет с успехом разъяснить смысл модели неспециалистам. До трех измерений есть возможно с успехом визуализировать документ и векторы запросов. На рисунке 5 изображено трехмерное пространство с визуализацией вектора документа (d) и вектора запроса (q), охватываемая тремя терминами Фамилия, Имя и Отчество.

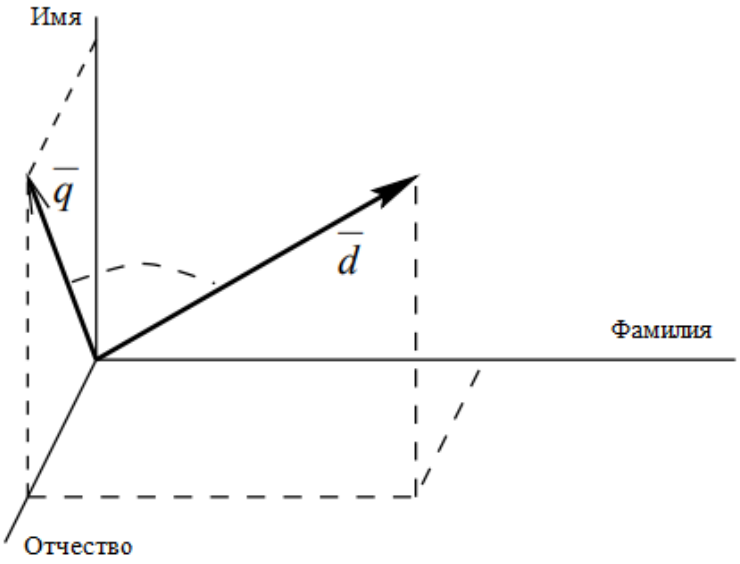


Рисунок 5 – Модель векторного пространство, где представляется документ и запрос

Представления и запросы должным образом нормализованы, тогда мера векторного произведения уравнения векторного внутреннего произведения действительно имеет сильную теоретическую мотивацию. Затем формула (3) изменяется, где условие обновленной формулы (4) и (5).

$$\text{score}(\vec{d}, \vec{q}) = \sum_{k=1}^{m} n(d_k) * n(q_k) \quad (3)$$

$$\text{n}(d_k) = \frac{d_k}{\sqrt{\sum_{k=1}^{m} (d_k)^2}} \quad (4)$$

$$\mathrm{n}(q_k) = \frac{q_k}{\sqrt{\sum_{k=1}^{m}(q_k)^2}} \quad (5)$$

Джурж Химста писал: «Основным недостатком модели векторного пространства является то, что она никоим образом не определяет, какими должны быть значения компонентов вектора. Проблема присвоения соответствующих значений компонентам вектора известна как взвешивание терминов. Ранние эксперименты Солтона в 1971 году и Солтона и Янга в 1973 году показали, что взвешивание терминов вовсе не является тривиальной проблемой. Они предложили так называемые веса tf.idf, комбинацию частоты терминов tf, которая представляет собой количество вхождений термина в документе, и idf , обратную частоту документа, которая представляет собой значение обратно пропорционально частоте документа df , которая представляет собой количество документов, содержащих этот термин. Многие современные алгоритмы взвешивания являются версиями семейства алгоритмов взвешивания tf.idf. Оригинальные веса Салтона tf.idf работают относительно плохо, в некоторых случаях хуже, чем простое взвешивание idf. Они определяются как веса tf.idf (6)»[20].

$$d_k = q_k = tf(k,d) * \log\frac{N}{df(k)} \quad (6)$$

где, t – число различных терминов в коллекции документов,

tf(k,d) - количество вхождений терминов k в документе d,

df(k) - количество документов, содержащих k,

N - общее количество документов в множестве.

Есть еще одна трудность, связанная с моделью векторного пространства – реализация модели векторного пространства, так как для вычисления косинусной меры необходимы значения всех векторных компонент, которые недоступны в инвертированном файле. Из-за этого нужно применять алгоритм векторного произведения и нормализованные

значения. Иначе нормализованные веса должны храниться в перевернутом файл, или значения нормализации должны храниться отдельно. Все это затруднительно в случае постепенного обновления индекса: добавление одного нового документа изменяет частоту терминов, встречающихся в документе, что изменяет длины векторов каждого документа, содержащего один или несколько из этих терминов.

1.2.3 Вероятностная модель

Любой документ завершается набором допустимых индексных терминов, взвешенных с помощью P(T|D), где P(T|D) - вероятность того, что, в случае если пользователь проявит желания получить информацию такого рода, которая содержится в документе D, он сформулирует запрос с помощью теоремы Байеса (7).

$$P(D|T) = \frac{P(T|D)*P(D)}{P(T)} \quad (7)$$

где, P(D|T) - вероятность того, что событие D истинно, если событие T истинно,

P(T|D) - вероятность того, что событие T истинно, если событие D истинно,

P(D) - вероятность того, что событие D истинно,

P(T) - вероятность того, что событие T истинно.

После необходимо ранжировать документы по P(D|T), то есть вероятности того, что документ D является релевантным, учитывая, что пользователь сформулировал запрос, используя термин T. Главное, что P(T) в знаменателе правой части является постоянным для любого заданного термина запроса T, и, следовательно, документы могут быть ранжированы по P(T|D)*P(D), что является величиной, пропорциональной значению P(D|T). В формуле P(D) - априорная вероятность релевантности документа D.

Принимая во внимание, что P(T|D) определяется индексатором, Марона и Кунса предполагают, что P(D) может быть определено статистикой использования документа, т.е. отношение количества использований документа D к общему количеству использований документа. Таким образом, их использование документа до P(D) можно рассматривать как самое первое описание рейтинга популярности, которое стало значительным для интернет-поиска. Любопытно, что оценка P(T|D) может быть получена аналогично, сохраняя для каждого использования документа также термин запроса, который был введен для извлечения документа в первую очередь.

Стивен Робертсон преобразовал введенную идею Марона и Кунса о ранжировании по вероятности релевантности в принцип. Он сформулировал принцип ранжирования вероятностей, который приписал Уильяму Куперу в 1977 году. Ван Рейсберген писал: «Если ответ справочной поисковой системы на каждый запрос представляет собой ранжирование документов в коллекциях в порядке уменьшения вероятности полезности для пользователя, отправившего запрос, где вероятности оцениваются как можно точнее на основе любых данных, которые были предоставлены системе для этой цели, тогда общая эффективность системы для ее пользователей будет наилучшей, которую можно получить на основе этих данных» [25:113].

Стивен Робертсон и Карен Спарк-Джонс основали свою вероятностную модель поиска на этой линии рассуждений. Они предложили ранжировать документы по P(R|D), то есть вероятности релевантности R дано описание содержимого документа D. Обратите внимание, что D здесь представляет собой вектор двоичных компонентов, каждый компонент обычно представляет термин, тогда как в предыдущем разделе D был «соответствующим документом». В вероятностном модель поиска вероятность P(R|D) интерпретируется следующим образом: может быть некоторое количество, допустим, 10 документов, которые представлены одним и тем же D. В случае если 9 из них считаются релевантными, то P(R|D) = 0,9. Чтобы это сработало на практике, используется правило Байеса

о коэффициентах вероятности P(R|D)/P(R|D), где R обозначает нерелевантность. Коэффициенты дают возможность избегать P(D) при вычислении, поддерживая при этом ранжирование по вероятности релевантности. Помимо этого, предполагается независимость между терминами, беря во внимание их актуальность.

Рассмотрим формула вероятностной модели поиска (8).

$$\frac{\mathrm{P(R|D)}}{\mathrm{P}(\overline{R}|\mathrm{D})} = \frac{\mathrm{P(D|R)P(R)}}{\mathrm{P(D}|\overline{R})\mathrm{P}(\overline{R})} = \frac{\prod_K \mathrm{P}(D_K|\mathrm{R})\mathrm{P(R)}}{\prod_K \mathrm{P}(D_K|\overline{R})\mathrm{P}(\overline{R})} \quad (8)$$

где, P(R|D) - вероятность того, что событие R истинно, если событие D истинно,

$\mathrm{P}(\overline{R}|\mathrm{D})$ - вероятность того, что событие R неистинно, если событие D истинно,

P(D|R) - вероятность того, что событие D истинно, если событие R истинно,

$\mathrm{P}(\overline{D}|\mathrm{R})$ - вероятность того, что событие D неистинно, если событие R истинно,

P(R) - вероятность того, что событие R истинно,

$\mathrm{P}(\overline{R})$ - вероятность того, что событие R неистинно,

$D_{\mathrm{к}}$ - k-й компонент (термин) в векторе документа

$\mathrm{P}(D_K|\mathrm{R})$ - вероятность того, что событие D_K истинно, если событие R истинно,

$\mathrm{P}(D_K|\overline{R})$ - вероятность того, что событие D_K истинно, если событие R неистинно.

Более удобная реализация вероятностного поиска использует следующие три преобразования с сохранением порядка.

Во-первых, документы ранжируются по суммам логарифмических коэффициентов, а не по самим коэффициентам. Во-вторых, априорные шансы релевантности P(R)/P(R) игнорируются. В-третьих, мы вычитаем

$\Sigma_к$log(P($D_к$=0|R)/P($D_к$= 0|$\bar{R}$)), т.е. оценка пустого документа из всех оценок документа. Таким образом, сумма по всем терминам, которая может составлять миллионы терминов, включает только ненулевые значения для терминов, присутствующих в документе. Улучшенная модель вероятностного поиска (9).

$$\sum \log \frac{P(D_k=1|R)*P(D_k=0|\overline{R})}{P(D_k=1|\overline{R})*P(D_k=0|\mathrm{R})} \quad (9)$$

где, $\mathrm{P}(D_K = 1|\mathrm{R})$ - вероятность того, что событие D_K истинно, если событие R истинно,

$P(D_k = 0|\overline{R})$ - вероятность того, что событие D_K неистинно, если событие R неистинно,

$P(D_k = 1|\overline{R})$- вероятность того, что событие D_K истинно, если событие R неистинно,

$\mathrm{P}(D_K = 1|\mathrm{R})$ - вероятность того, что событие D_K истинно, если событие R истинно.

Джурж Химста писал: «На практике термины, которых нет в запросе, также игнорируются в данном уравнении. Для полного использования модели вероятностного поиска требуются две вещи: примеры соответствующих документов и длинные запросы. Соответствующие документы необходимы для вычисления P($D_к$|R), то есть вероятности того, что документ содержит термин k с учетом релевантности. Длинные запросы необходимы, поскольку модель различает только наличие термина и отсутствие термина в документах, и, как следствие, количество различных значений оценок документа невелико для коротких запросов. Для одного-единственного слова запроса, число различных вероятностей равно двум (либо документ содержит слово, либо нет), для запроса из двух слов оно равно четырем (документ содержит оба термина, или только первый термин, или только второй, или ни один), для запроса из трех слов оно равно восьми

и т.д. Очевидно, что это делает модель неадекватной для Web-поиска, для которого заранее не известны соответствующие документы и для которого запросы, как правило, короткие. Однако эта модель полезна, например, в фильтрах нежелательной почты. Спам-фильтры накапливают множество примеров релевантных и нерелевантные (спам) документы с течением времени. Чтобы определить, является ли входящее электронное письмо спамом вместо нескольких терминов запроса можно использовать полный текст электронного письма» [20]. Значит, что для вероятностной модели требуются примеры релевантных и нерелевантных документов, чтобы использовать ее для Web-поиска. Соответственно, в моем случае, эта модель не подходит для Web-приложения для поиска событий в базе СКУД.

Вывод по разделу 1

Таким образом, благодаря анализу трех математических моделей можно отметить, что в первом разделе анализируются 3 математические модели. В информационном поиске одни модели работают для одних приложений, тогда как другие работают для других приложений. Например, булева модель в разделе 1.2.1 отлично подойдет для релевантных документов; модели векторного пространства в разделе 1.2.2 хорошо подходят для поиска сходства и обратной связи по релевантности во многих ситуациях, если доступна хорошая весовая функция; вероятностная модель поиска из раздела 1.2.3 может быть хорошим выбором, если доступны примеры релевантных и нерелевантных документов. В моем случае отлично подойдет булева модель, ведь данные в базе данных хранятся в именительном падеже, то есть в нормальной форме.

2 Проектирование и разработка Web-приложения

2.1 Обоснование выбора средств разработки

Исходными данными к выпускной квалификационной работе является тестовая база APACS 3000, которая аналогична той, в которой ведется действительный учёт событий в баз СКУД ТГУ и система управления базами данных Firebird.

APACS 3000 – это универсальный программный комплекс для управления системами безопасности, контроля доступа и учета рабочего времени. Уникальность APACS 3000 заключается в том, что он одинаково эффективно работает на объектах любых типов и масштабов. Это возможно благодаря его масштабируемости и различным модификациям, позволяющем не переплачивать за неиспользуемый функционал. База данных APACS 3000 построена на среде Firebird.

Firebird - это бесплатная система управления реляционными базами данных SQL с открытым исходным кодом, основанная на версии InterBase с открытым исходным кодом, выпущенной Borland Software Corp, ранее известной как Inprise Corp. Разработанный на C и C ++, Firebird поддерживает основные аппаратные и программные платформы, включая Windows, Linux и Mac OS X. Firebird предлагает множество стандартных функций ANSI SQL, позволяет одновременные операции OLTP и OLAP через свою многопоколенную архитектуру и поддерживает хранимые процедуры и триггеры.

Для разработки и администрирования СУБД (Firebird) была использована GUI-оболочка — IBExpert. GUI — это графический интерфейс пользователя. Графический интерфейс пользователя — это форма пользовательского интерфейса, которая позволяет пользователям взаимодействовать с электронными устройствами с помощью графических значков и аудио-индикатора, таких как первичная нотация, вместо текстовых

пользовательских интерфейсов, набранных командных меток или текстовой навигации. На рисунке 6 наглядно представлено взаимодействие между тестовой базой данных APACS 3000, СУБД — Firebird и приложением баз данных — IBExpert [2].

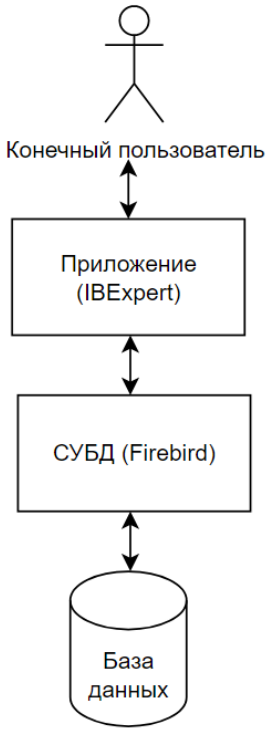


Рисунок 6 – Схематичное представление компонент системы баз данных

Для в Web-приложения был выбран язык PHP, так как он наиболее часто используется для этих целей. PHP — это серверный язык сценариев, разработанный специально для Web-разработки. Инфраструктура PHP с открытым исходным кодом, что означает, что его можно бесплатно скачать и использовать [8], [9], [15], [19]. Проект W3Techs австрийской компании Q-Success Web-based Services собирает статистику использования Web-технологий. По их данным, опубликованных 1 мая 2022 года, видно, что 77,5% сайтов, о бэкенде которых известно, используют PHP. На рисунке 7 представлена диаграмма, где указаны самые популярные языки серверного

программирования. Компании, которые используют PHP: WhatsApp, Lyft, Mint и Viber. В связи с этим на рынке труда Web-разработчик, работающий на языке PHP наиболее востребован [27].

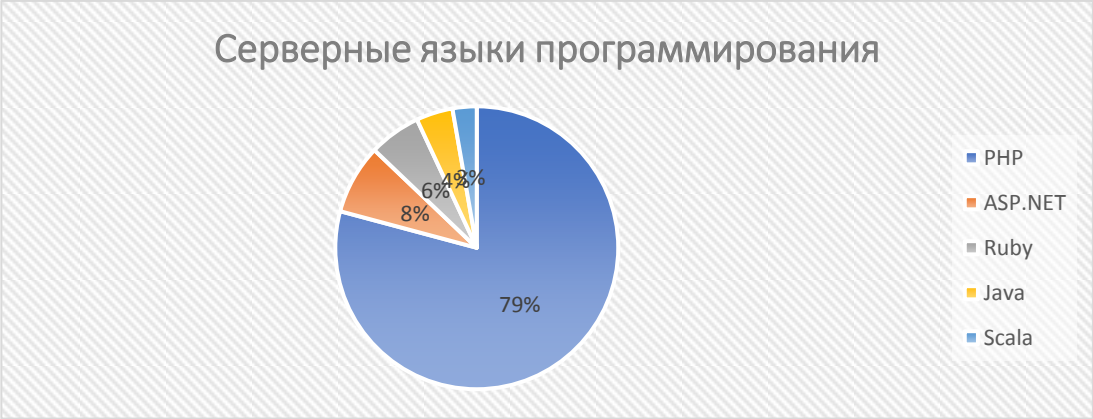


Рисунок 7 – Серверные языки программирования

Web-приложение удобнее и качественнее писать с помощью фреймворков. Мой выбор пал на Laravel, так как он один из самых популярных PHP фреймворков.

Структура кода Laravel соответствует общеизвестному паттерну проектирования MVC, у которого схема разделения данных приложения и управляющей логики на 3 отдельных компонента: модель, представление и контроллер.

Данный шаблон проектирования, который представлен на рисунке 8, позволяет отделить логику приложения от его визуальной части, что позволяет делать код более читабельным, а процесс разработки комфортным, разграничивая работу frontend- и backend-разработчиков.

Также среди вакантных мест на данный момент Web-разработчиков на PHP часто требуется знание Laravel. Для его установки потребуется Composer [20], [23].

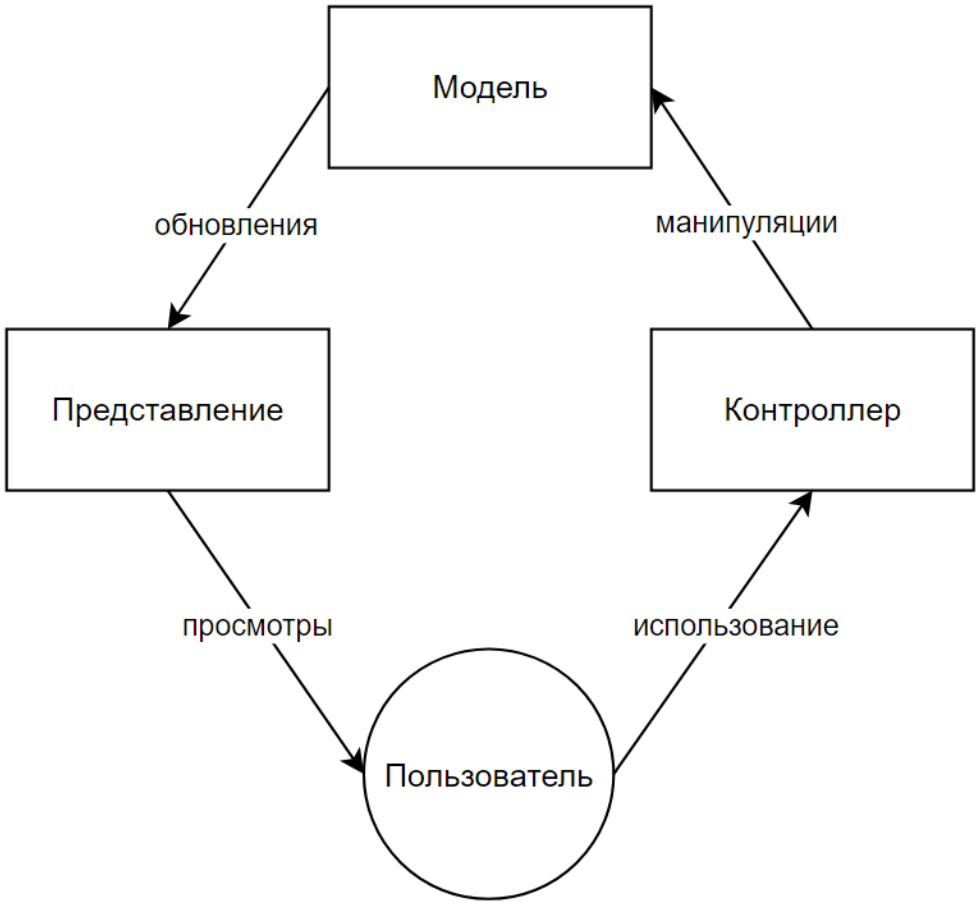


Рисунок 8 – Схематичное представление MVC

Для реализации зависимостей Laravel используют Composer, который автоматически загружается один раз и становится доступным для всей системы. Он помогает клиенту создать проект по отношению к его указанной системе и проекту. Сторонние библиотеки могут быть установлены в проекте без особых усилий с использованием Composer. Установка Laravel выполняется с помощью команды, выполненной в терминале: composer global require "laravel/installer=~1.1".

Далее представлен рисунок 9, на котором изображены используемые директории в проекте на Laravel.

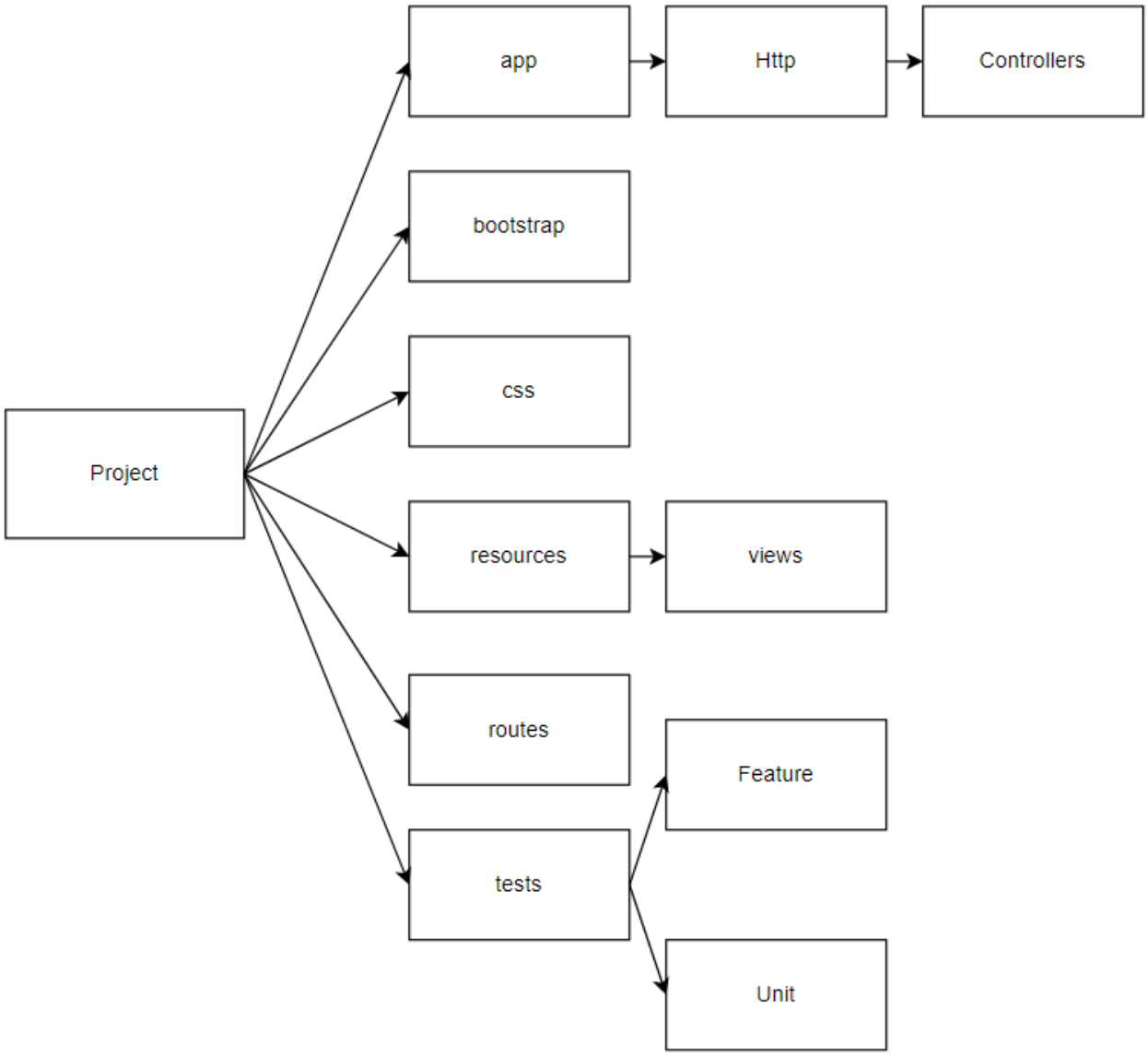


Рисунок 9 – Схематичное представление директорий, используемых в ходе разработки проекта.

Из рисунка 9 видно, что были использованы директории: Controllers, bootstrap, css, views, routes, Feature, Unit [5].

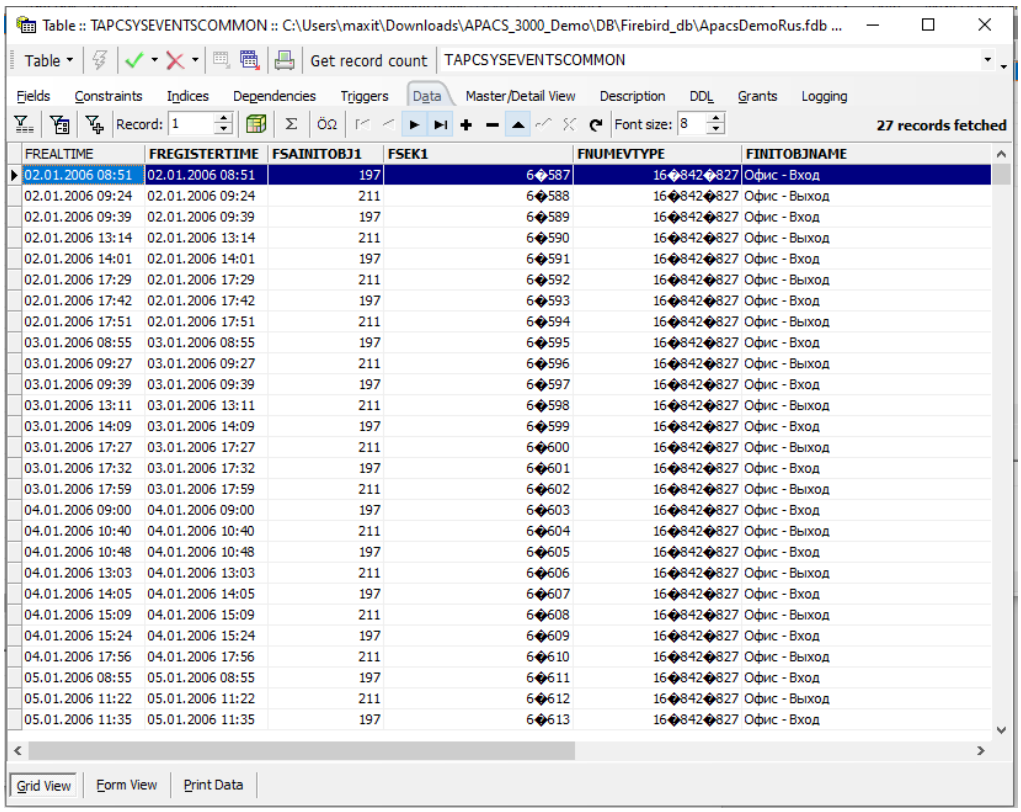
2.2 Разработка и реализация Web-приложения

Для работы с тестовой базой данных APACS_3000_Demo необходимо подключиться к ней в Laravel проекте как показано на рисунке 10 [18][22].

```
$dsn = 'firebird:dbname=C:\Users\maxit\Downloads\APACS_3000_Demo\DB\Firebird_db\ApacsDemoRus.FDB;http://localhost:4000;charset=WIN1251;';
$username = 'SYSDBA';
$password = 'masterkey';
$dbh= new \PDO($dsn, $username, $password, [\PDO::ATTR_ERRMODE => \PDO::ERRMODE_EXCEPTION]);
```

Рисунок 10 – Подключение к базе данных

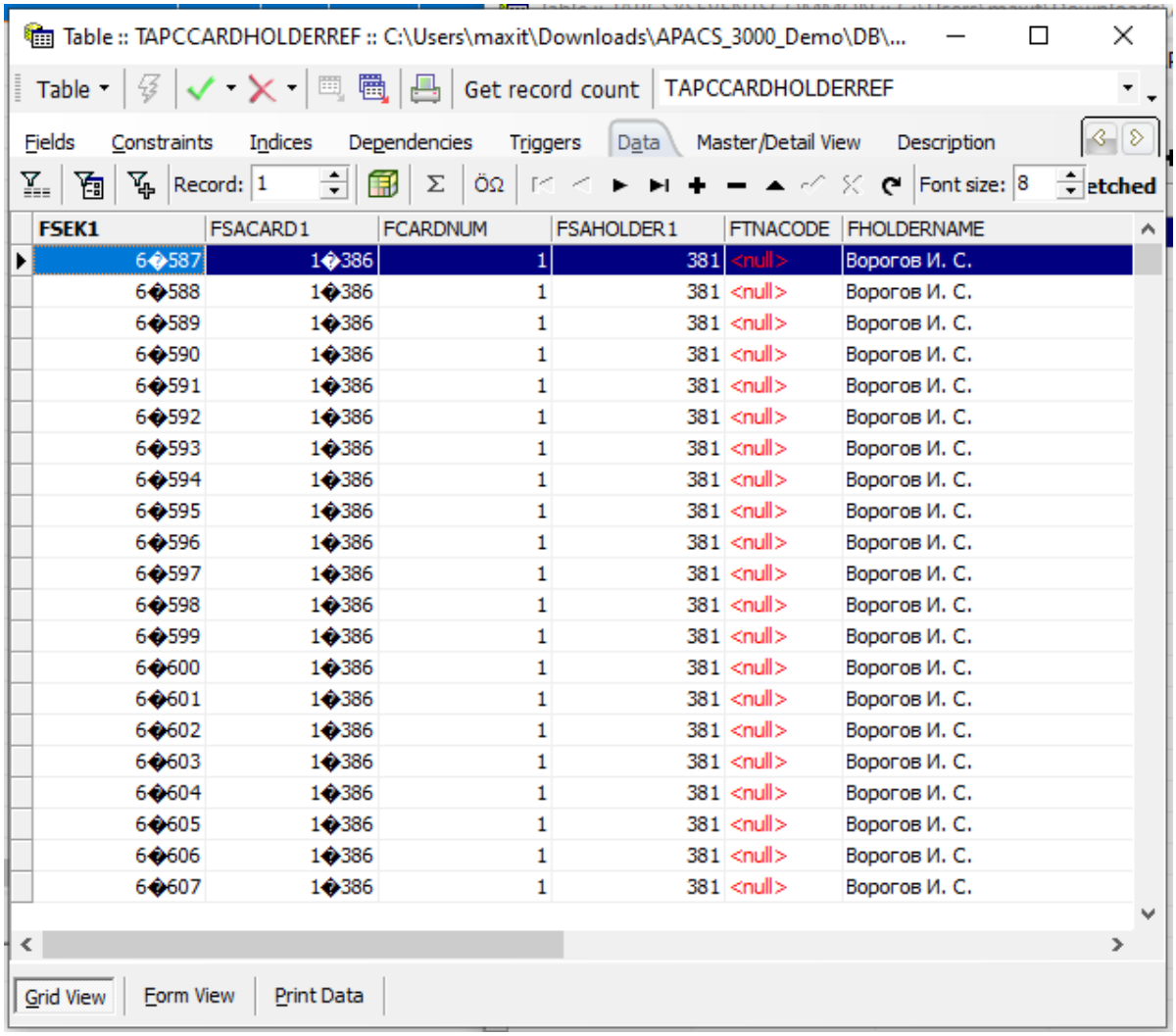
Из тестовой базы APACS_3000_Demo были задействованы 3 таблицы из 217: TAPCSYSEVENTSCOMMON, TAPCCARDHOLDERREF, TAPCCARDHOLDER, так как именно в них хранится информация о событиях и персональных данных. Из таблицы TAPCSYSEVENTSCOMMON необходимы столбцы с обозначением времени события: FREALTIME, использованного роторного турникета: FINITOBJNAME и номера события: FSEK1. Фрагмент TAPCSYSEVENTSCOMMON показан на рисунке 11.



FREALTIME	FREGISTERTIME	FSAINITOBJ1	FSEK1	FNUMEVTYPE	FINITOBJNAME
02.01.2006 08:51	02.01.2006 08:51	197	6€587	16€842€827	Офис - Вход
02.01.2006 09:24	02.01.2006 09:24	211	6€588	16€842€827	Офис - Выход
02.01.2006 09:39	02.01.2006 09:39	197	6€589	16€842€827	Офис - Вход
02.01.2006 13:14	02.01.2006 13:14	211	6€590	16€842€827	Офис - Выход
02.01.2006 14:01	02.01.2006 14:01	197	6€591	16€842€827	Офис - Вход
02.01.2006 17:29	02.01.2006 17:29	211	6€592	16€842€827	Офис - Выход
02.01.2006 17:42	02.01.2006 17:42	197	6€593	16€842€827	Офис - Вход
02.01.2006 17:51	02.01.2006 17:51	211	6€594	16€842€827	Офис - Выход
03.01.2006 08:55	03.01.2006 08:55	197	6€595	16€842€827	Офис - Вход
03.01.2006 09:27	03.01.2006 09:27	211	6€596	16€842€827	Офис - Выход
03.01.2006 09:39	03.01.2006 09:39	197	6€597	16€842€827	Офис - Вход
03.01.2006 13:11	03.01.2006 13:11	211	6€598	16€842€827	Офис - Выход
03.01.2006 14:09	03.01.2006 14:09	197	6€599	16€842€827	Офис - Вход
03.01.2006 17:27	03.01.2006 17:27	211	6€600	16€842€827	Офис - Выход
03.01.2006 17:32	03.01.2006 17:32	197	6€601	16€842€827	Офис - Вход
03.01.2006 17:59	03.01.2006 17:59	211	6€602	16€842€827	Офис - Выход
04.01.2006 09:00	04.01.2006 09:00	197	6€603	16€842€827	Офис - Вход
04.01.2006 10:40	04.01.2006 10:40	211	6€604	16€842€827	Офис - Выход
04.01.2006 10:48	04.01.2006 10:48	197	6€605	16€842€827	Офис - Вход
04.01.2006 13:03	04.01.2006 13:03	211	6€606	16€842€827	Офис - Выход
04.01.2006 14:05	04.01.2006 14:05	197	6€607	16€842€827	Офис - Вход
04.01.2006 15:09	04.01.2006 15:09	211	6€608	16€842€827	Офис - Выход
04.01.2006 15:24	04.01.2006 15:24	197	6€609	16€842€827	Офис - Вход
04.01.2006 17:56	04.01.2006 17:56	211	6€610	16€842€827	Офис - Выход
05.01.2006 08:55	05.01.2006 08:55	197	6€611	16€842€827	Офис - Вход
05.01.2006 11:22	05.01.2006 11:22	211	6€612	16€842€827	Офис - Выход
05.01.2006 11:35	05.01.2006 11:35	197	6€613	16€842€827	Офис - Вход

Рисунок 11 – Фрагмент заполненных данных в таблице TAPCSYSEVENTSCOMMON

Из таблицы TAPCCARDHOLDERREF необходимы столбцы с обозначением номера события: FSEK1 и номеров объекта: FSAHOLDER1. Фрагмент TAPCCARDHOLDER показан на рисунке 12.



FSEK1	FSACARD1	FCARDNUM	FSAHOLDER1	FTNACODE	FHOLDERNAME
6◆587	1◆386	1	381	<null>	Воронов И. С.
6◆588	1◆386	1	381	<null>	Воронов И. С.
6◆589	1◆386	1	381	<null>	Воронов И. С.
6◆590	1◆386	1	381	<null>	Воронов И. С.
6◆591	1◆386	1	381	<null>	Воронов И. С.
6◆592	1◆386	1	381	<null>	Воронов И. С.
6◆593	1◆386	1	381	<null>	Воронов И. С.
6◆594	1◆386	1	381	<null>	Воронов И. С.
6◆595	1◆386	1	381	<null>	Воронов И. С.
6◆596	1◆386	1	381	<null>	Воронов И. С.
6◆597	1◆386	1	381	<null>	Воронов И. С.
6◆598	1◆386	1	381	<null>	Воронов И. С.
6◆599	1◆386	1	381	<null>	Воронов И. С.
6◆600	1◆386	1	381	<null>	Воронов И. С.
6◆601	1◆386	1	381	<null>	Воронов И. С.
6◆602	1◆386	1	381	<null>	Воронов И. С.
6◆603	1◆386	1	381	<null>	Воронов И. С.
6◆604	1◆386	1	381	<null>	Воронов И. С.
6◆605	1◆386	1	381	<null>	Воронов И. С.
6◆606	1◆386	1	381	<null>	Воронов И. С.
6◆607	1◆386	1	381	<null>	Воронов И. С.

Рисунок 12 – Фрагмент заполненных данных в таблице TAPCCARDHOLDERREF

Из таблицы TAPCCARDHOLDER необходимы столбцы с обозначением номера объекта: FID1, фамилии: FLASTNAME, имени: FFIRSTNAME и отчества: FMIDDLENAME. Фрагмент TAPCCARDHOLDER показан на рисунке 13.

Из таблиц TAPCSYSEVENTSCOMMON, TAPCCARDHOLDERREF и столбцов FSEK1 и FSEK1 формируется единая таблица с номерами событий с помощью команды inner join [11][12].

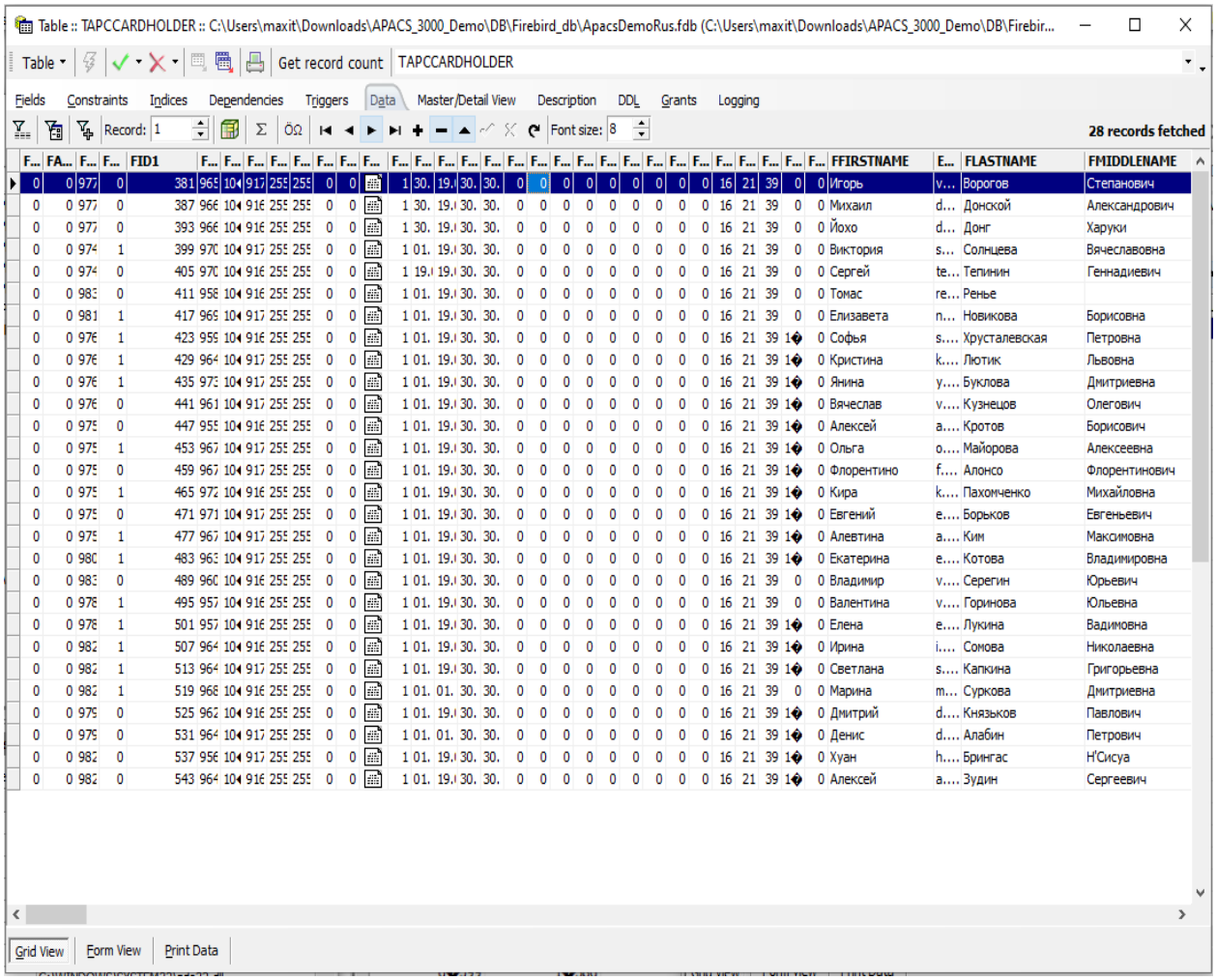


Рисунок 13 – Фрагмент заполненных данных в таблице TAPCCARDHOLDER

Далее из таблиц TAPCCARDHOLDERREF, TAPCCARDHOLDER и столбцов FSAHOLDER1 и FID1 формируется еще одна единая таблица с номерами объектов. INNER JOIN объединяет записи из двух таблиц, если в связующих полях этих таблиц содержатся одинаковые значения, как показано на рисунке 14 [11], [12].

В запросе $sql переменные $startDate и $endDate являются динамическими, как и $lastName, $firstName, $middleName.

```
$sql="SELECT FREALTIME, FINITOBJNAME, FHOLDERNAME, FFIRSTNAME, FLASTNAME, FMIDDLENAME  FROM TAPCSYSEVENTSCOMMON
INNER JOIN tapccardholderref ON (tapcsyseventscommon.fsek1 = tapccardholderref.fsek1)
INNER JOIN tapccardholder ON (tapccardholderref.fsaholder1 = tapccardholder.fid1)
WHERE FREALTIME >= '" . $startDate . "' and FREALTIME <= '" . $endDate ."' ";

if ($lastName) {
 $sql = $sql  . " AND  FLASTNAME = '" . $lastName . "'";
}

if ($firstName) {
 $sql = $sql  . " AND  FFIRSTNAME= '" . $firstName . "'";
}

if ($middleName) {
 $sql = $sql  . " AND  FMIDDLENAME = '" . $middleName . "'";
}
```

Рисунок 14 – Запрос с использованием таблиц TAPCCARDHOLDER, TAPCCARDHOLDERREF, TAPCSYSEVENTSCOMMON для поиска по фамилии, имени, отчеству и временному промежутку

На главной странице Web-приложения можно увидеть поля для ввода значений для данных переменных, как показано на рисунке 15 [1], [7], [17], [24], [26].

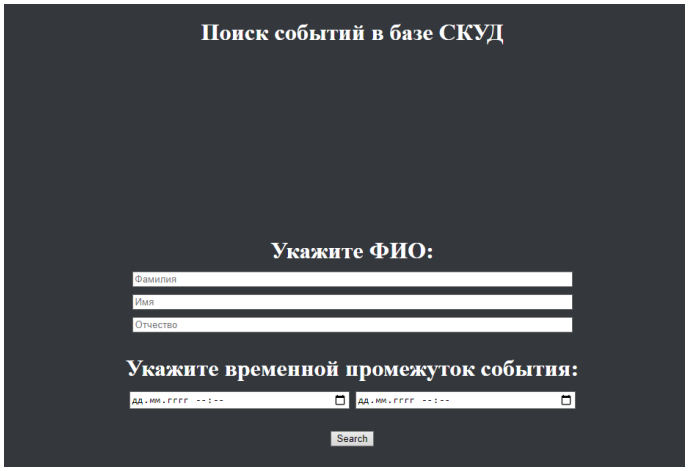


Рисунок 15 – Внешний вид главной страницы Web-приложения

Для интерактивных полей ввода и кнопки в шаблоне search.blade.php написан код. Шаблонизатор, который помогает разграничить HTML и

сделать работу с ним и вывод php данных более удобным и гибким [10][16][21]. Он основан на концепции наследования шаблонов и секциях, представлен на рисунке 16.

resources > views > search.blade.php

```
@extends('layout')
@section('main_content')
<body>
    <h1>Поиск событий в базе СКУД<h1>
        <div id="wrapper" ><br><br><br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
        <form  method="get" action="{{ route ('search') }}" >

        <label for="name">Укажите ФИО:</label> <br><br>
      <input  type="text" id ="lastName" name="lastName" placeholder="Фамилия" style="width:60%" > <br><br>
      <label for="name"></label>
      <input  type="text" id ="firstName" name="firstName" placeholder="Имя" style="width:60%" > <br><br>
      <label for="name"></label>
      <input  type="text" id ="middleName" name="middleName"placeholder="Отчество" style="width:60%" > <br><br>
      <br><br>

    <label for="party">Укажите временной промежуток события:</label><br>

    <input id="starDate" type="datetime-local" name="startDate"
           min="2006-01-01T08:30" max="2022-01-30T16:30" style="width:30%"
           pattern="[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}" required>
    <input id="endDate" type="datetime-local" name="endDate"
           min="2006-01-01T08:30" max="2022-02-30T16:30" style="width:30%"
           pattern="[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}" required>
    <br><br> <br><br>

    <input  type="submit" name="submit" value="Search">

    </form>
    <div>
  </body>
```

Рисунок 16 – Код resources\views\search.blade.php

Данный шаблон наследован от другого шаблона resources\views\layout.blade.php, Директива @section определяет секцию содержимого [10], [16], [21]. Как показано на рисунке 17.

В шаблоне resources\views\layout.blade.php директива @yield используется для отображения содержимого заданной секции. Т.е. в то место, где стоит @yield попадает значение секции, в котором хранится содержимое о CSS, то есть о каскадных таблицах стилей. Они представляют собой набор параметров форматирования, который применяется к элементам документа, чтобы изменить их внешний вид.

resources > views > layout.blade.php

```
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>Поиск событий</title>

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

</head>

<body>
@yield('main_content')
</body>
</html>
```

Рисунок 17 – Код resources\views\layout.blade.php

На рисунке 18 представлен набор параметров форматирования, который применяется к элементам документа. В моем случае они были пригодны для форматирования отступов, цветовых параметров, централизации текста [3], [10], [13], [16].

css > # style.css > ...

```
br {
    line-height: 10px;
}
body{
    background: rgb(52, 56, 61);
}
h1{
    color: rgb(255, 255, 255);
    margin: 20px auto 0;
    width: 1000px;
    text-align: center;
}
```

Рисунок 18 – Код public\style.css

В файле CSS определяется способ отображения фона Web-приложения, а также централизация текста и размер отступов [6], [21].

Далее представлены экранные формы на рисунках 19-24 результатов запросов с поиском по ФИО и временному промежутку, реализованные в Web-приложении.

На следующей экранной форме, то есть на рисунке 19, можно увидеть результат запроса по временному промежутку времени с 01.01.2006 00:00 по 01.01.2009 00:00.

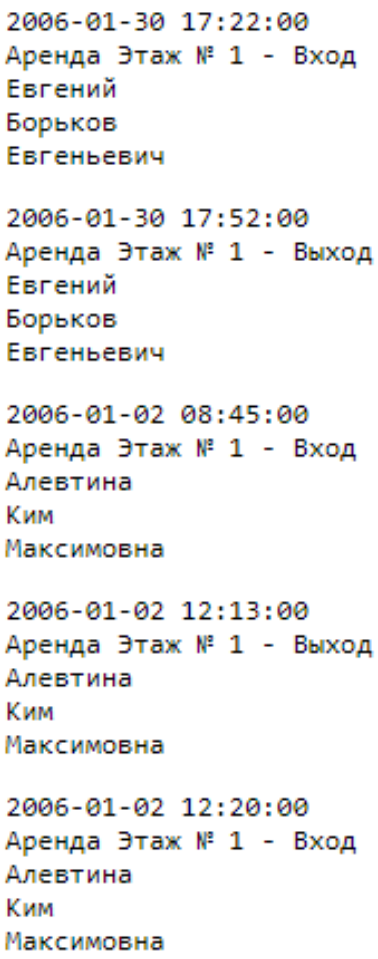


Рисунок 19 – Фрагмент результата по запросу временной промежуток с 01.01.2006 00:00 по 01.01.2009 00:00

На следующей экранной форме, то есть на рисунке 20, можно увидеть запрос по временному промежутку времени с 01.01.2006 00:00 по 01.01.2009 00:00.

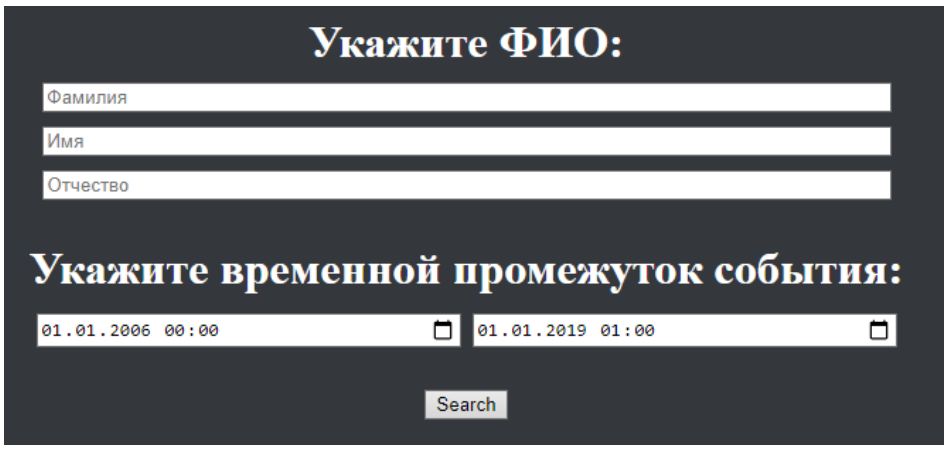


Рисунок 20 – Временной промежуток с 01.01.2006 00:00 по 01.01.2009 00:00

Теперь проверим работу запроса с помощью запроса по имени Алексей, которое встречается в базе данных APACS_3000_Demo в двух объектах, представлена экранная форма запроса на рисунке 21.

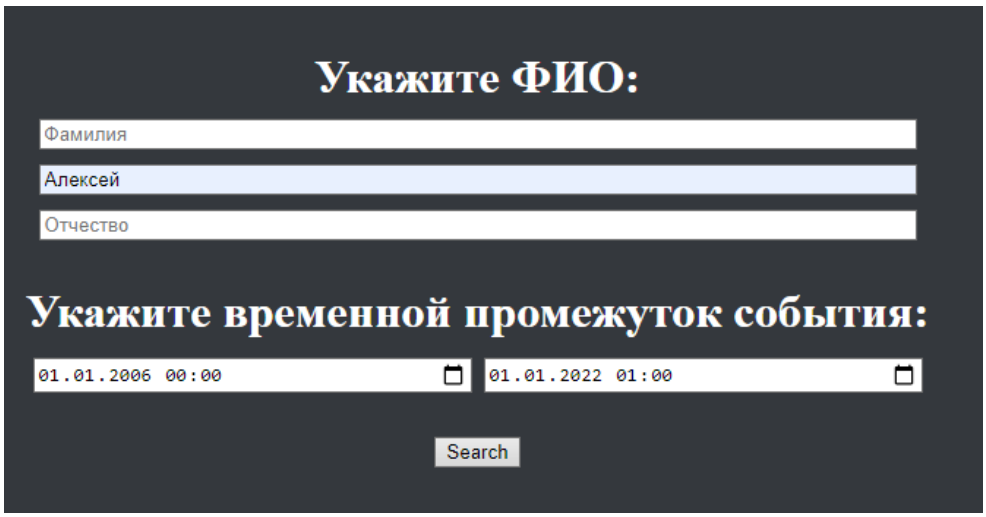


Рисунок 21 – Запрос с указанием имени Алексей

На следующей экранной форме видно, что запрос выполнился корректно, так как 2 объекта включены в выборку. Первый – «Алексей Кротов Борисович». Второй – «Алексей Зудин Сергеевич», показано на рисунке 22.

```
2006-01-30 18:04:00
Аренда Этаж № 1 - Выход
Алексей
Кротов
Борисович

2006-01-02 09:04:00
Офис - Вход
Алексей
Зудин
Сергеевич

2006-01-02 10:58:00
Офис - Выход
Алексей
Зудин
Сергеевич

2006-01-02 11:01:00
Офис - Вход
Алексей
Зудин
Сергеевич

2006-01-02 13:05:00
Офис - Выход
Алексей
Зудин
Сергеевич
```

Рисунок 22 – Фрагмент результата запроса, выводящих список с именем Алексей

Рассмотрим запрос, включающий в себя выборку и по имени, и по временному промежутку, представленный на рисунке 23.

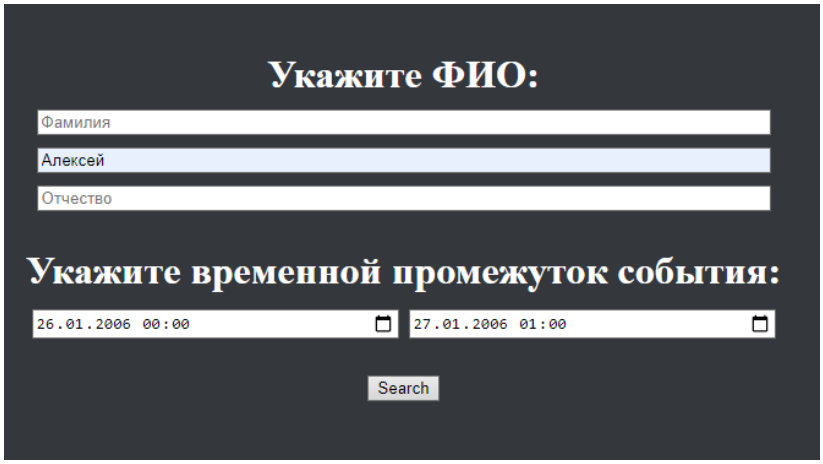


Рисунок 23 – Результат запроса по имени Алексей и по числу 26 января 2006 года

На следующей экранной форме видно, что запрос выполнился корректно, так как дата соответствует дате, указанной в запросе, а также соблюден критерий по имени, представленной на рисунке 24.

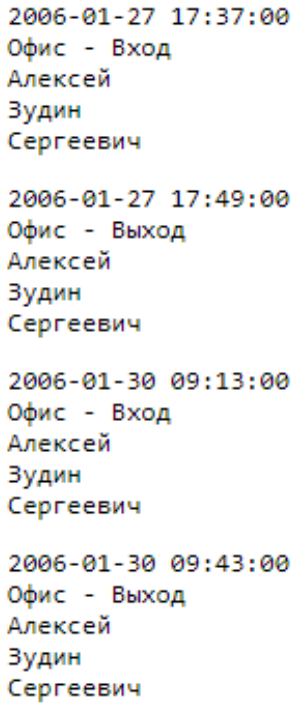


Рисунок 24 – Результат запроса по имени Алексей и по числу 26 января 2006 года

Помимо этого, можно указывать в запросе только фамилию или отчество. Также при желании запрос может быть заполнен полностью и при этом корректно выполняться.

2.3 Тестирование Web-приложения

Результатом тестирования приложения должна стать корректная работа программы.

Для проведения тестирования в фреймворке Laravel по умолчанию добавлена директория – test. В ней есть две папки: Feature и Unit. В каждой

из них сгенерирован код на php, где описаны тестовые классы. На рисунке 25 представлен код ExampleTest.php из папки Feature.

tests > Feature > ExampleTest.php

```php
<?php

namespace Tests\Feature;

use Illuminate\Foundation\Testing\RefreshDatabase;
use Tests\TestCase;

class ExampleTest extends TestCase
{
    /**
     * A basic test example.
     *
     * @return void
     */
    public function test_the_application_returns_a_successful_response()
    {
        $response = $this->get('/');

        $response->assertStatus(200);
    }
}
```

Рисунок 25 – Код базового теста ExampleTest.php из папки Feature

На рисунке 25 видно, что функция рассматривает главную страницу, где статус должен быть равен 200, что означает успешное тестирование, также обозначается как OK.

Также на рисунке 26 можно увидеть код ExampleTest.php из папки Unit.

ExampleTest.php проверяет работоспособность самого тестирования, то есть правильность его выполнения.

```
tests > Unit > ExampleTest.php
1   <?php
2
3   namespace Tests\Unit;
4
5   use PHPUnit\Framework\TestCase;
6
7   class ExampleTest extends TestCase
8   {
9       /**
10       * A basic test example.
11       *
12       * @return void
13       */
14      public function test_that_true_is_true()
15      {
16          $this->assertTrue(true);
17      }
18  }
```

Рисунок 26 - Код базового теста ExampleTest.php из папки Unit

Для проведения тестирования нужно выполнить команду в терминале: php artisan test. С результатами можно ознакомиться на рисунке 27.

```
PASS  Tests\Unit\ExampleTest
✓ that true is true

PASS  Tests\Feature\ExampleTest
✓ the application returns a successful response

Tests:  2 passed
Time:   0.14s
```

Рисунок 27 – Результаты тестирования

Так как два тестирования завершились с успехом, можно создать свой тестовый файл, в котором будут проверяться страницы Web-приложения. Одна из страниц – это главная страница, а вторая – это результат полученных событий.

Для создания своего тестового файла используется команда в терминале: php artisan make:test ProjectTest –unit. Далее там пишется код. Код файла ProjectTest можно увидеть на рисунке 28.

tests > Unit > ProjectTest.php

```php
<?php

namespace Tests\Unit;

use Tests\TestCase;

class ProjectTest extends TestCase
{

    public function test_example_test()
    {
        $responce=$this->get('/');
        $responce->assertOk();
    }

    public function test_example_test_2()
    {
        $responce=$this->get('/events');
        $responce->assertOk();
    }


}
```

Рисунок 28 – Код теста ProjectTest

С результатами данного теста, а также двух предыдущих можно ознакомиться на рисунке 29.

```
PASS  Tests\Unit\ExampleTest
✓ that true is true

PASS  Tests\Unit\ProjectTest
✓ example test
✓ example test 2

PASS  Tests\Feature\ExampleTest
✓ the application returns a successful response

Tests:  4 passed
Time:   0.31s
```

Рисунок 29 – Результаты тестирования

В результате тестирования были получены положительные результаты. Два базовых теста: ExampleTest из папок Feature и Unit успешно выполнены. Также созданный мной тест ProjectTest, в котором проверяются две страницы Web-приложения, где первая страница – это главное меню с полями для ввода фамилии, имени, отчества, а также дат, а вторая страница – это результат с описанием событий по заданному запросу, выполняется благоприятно.

Вывод по разделу

Было реализовано Web-приложение для поиска событий в базе СКУД ТГУ. В приложении реализован функционал поиска по фамилии, имени, отчеству и временному промежутку. Приложение было сделано с применением базы тестовой APACS 3000, которая построена на среде Firebird. GUI-оболочка, предназначенная для разработки и администрирования базs данных Firebird – IBExpert. Фреймворк был использован - Laravel. Скриптовый зык - PHP. Стандартизированный язык разметки – HTML. Формальный язык описания внешнего вида документа – CSS.

В разделе подробно описаны используемые таблицы из базы данных, описан запрос, с помощью которого осуществляется поиск по базе данных, а далее данные из базы данных передаются уже в Web-приложение, которое отображает релевантные данные пользователю.

В результате тестирования Web-приложения для поиска по событиям в базе СКУД было выявлено, что работа Web-приложения выполняется корректно.

Заключение

Итогом выпускной квалификационной работы являлось построение модели поиска событий в базе СКУД и апробация ее результативности через веб-приложение.

Была проанализирована характеристика объекта исследования, которым являются математические модели поиска. Выяснена причина создания моделей поиска информации – модели могут служить основой для реализации реальной информационно-поисковой системы. Кроме того, построена схема для наглядного представления основных процессов информационно-поисковой системы: представление содержимого документов, представление информационных потребностей пользователя и сравнение двух представлений.

В данной выпускной квалификационной работе были описаны три ключевых моделей поиска: булева, векторная, вероятностная. У каждой модели оказались свои преимущества и недостатки, поэтому был проведен сравнительный анализ математических моделей поиска.

Путем анализа математических моделей поиска было выявлено, что булева модель подойдет отлично для информационного поиска в приложении, которое осуществляет поиск по базе данных.

Был обоснован выбор средств разработки, а именно языка программирования - PHP, фреймворка - Laravel, каскадных таблиц стилей - CSS, язык разметки - HTML, тестовая база СКУД - APACS 3000, утилита, предназначенная для администрирования баз данных - IBExpert, реляционная система управления базами данных - Firebird. Схематично представлены используемые директории и модель MVC.

Были подробно описана работа в базе данных Firebird с данными, которые взяты из тестовой базы СКУД APACS 3000, а также реализация запроса по фамилии, имени, отчеству и датам для Web-приложения. После

проведено тестирование Web-приложения с с помощью трех различных тестов, которые два из них базовые, а третий – реализован мной.

Актуальность моей выпускной бакалаврской работы состоит в том, что Web-приложение, можно использовать локально, то есть не требуется установка на каждый ПК в организации «Информационный отдел безопасности ТГУ» или другой организацией, которая также использует APACS 3000.

Список используемой литературы

1. Астахова И.Ф. SQL в примерах и задачах: учебное пособие / И.Ф. Астахова, А.П. Толстобров, В.М. Мельников – Минск: Новое знание, 2002. – 176 с.
2. Бондарь А.Г. InterBase и Firebird. Практическое руководство для умных пользователей и начинающих разработчиков / А.Г. Бондарь – СII6.: БХВ-Петербург, 2012. – 610 с.
3. Дронов В. А. Разработка современных Web-сайтов / В. А. Дронов - СПб.: БХВ-Петербург, 2013. – 414 с.
4. Егорова А. Информационные системы: методы и средства проектирования [Электронный ресурс]. / А. Егорова – Московский государственный технический институт, 2006. – Режим доступа: https://cyberleninka.ru/article/n/informatsionnye-sistemy-metody-i-sredstvaproektirovaniya
5. Иванова, Г.С. Технология программирования: учебник/ Г.С. Иванова – М.: КНОРУС, 2011. – 336 с.
6. Киселев С. В. Веб-дизайн / С. В. Киселев, С. В. Алексахин, А. В. Остроух — Москва, Академия, 2019 г.- 64 с.
7. Колисниченко Д. Н. PHP и MySQL. Разработка веб-приложений / А.Г. Бондарь – 6-е изд.- СII6.: БХВ-Петербург, 2017. – 640 с.
8. Котеров Д. В. PHP 7 / Д. В. Котеров, И. В. Симдянов — Санкт-Петербург: БХВ-Петербург, 2019 — 1088 с.
9. Котеров Д. «PHP. В подлиннике» / Д. Котеров, А. Костарев — Спб.: «БХВ-Петербург», 2018, 1120 с.
10. Кожемякин А. А. HTML и CSS в примерах. Создание Web-страниц / А. А. Кожемякин - М.: Альтекс-А, 2014. – 416 с.
11. Кузин А.В. Базы данных: Учебное пособие для студентов высших учебных заведений / А.В. Кузин, С.В. Левонисова. – 5-е изд. — М.: ИЦ Академия, 2018. - 320 с.

12. Малыхина М. Базы данных. Основы, проектирование, использование [Текст] / М. Малыхина – БХВ-Петербург, 2006. – 528 с.

13. Печников В.Н. Создание Web-страниц и Web-сайтов. Самоучитель. / - М.: Триумф, 2013. - 470 с

14. Петухов О. А. Моделирование системное, имитационное, аналитическое: учебное пособие / О. А. Петухов, А. В. Морозов, Е. О. Петухова. —Ставрополь: Северо-Кавказский федеральный университет, 2008. — 276 с.

15. Руководство по PHP [Электронный ресурс] — Заглавие с экрана. Режим доступа: http://www.php.net/manual/ru/

16. Семикопенко А.А. Учебник HTML / А.А. Семикопенко – СПб.: БХВ-Петербург, 2017. - 190 с.

17. Симонов Д. Руководство по языку SQL СУБД Firebird 2.5: Firebird 2.5.9. v.0500-1 / Д. Симонов, П. Винкенуг, Д. Филиппов, Д. Еманов, А. Карпейкин, Д. Кузьменко, А. Ковязин — Москва: Московская биржа, IBSurgeon, 2020. — 576 с.

18. Симонов Д. Пакет для работы с СУБД Firebird в Laravel / Д. Симонов [Текст]. — Москва, 2016. — 9 с.

19. Bierer D. PHP 8 Programming Tips, Tricks and Best Practices: A practical guide to PHP 8 features, usage changes, and advanced programming techniques / D. Bierer, C. Evans —Packt Publishing, 2021 — 528 с.

20. Hiemstra D. Information Retrieval: Searching in the 21st Century / D. Hiemstra J. Davies, A. Goker, M. Graham — Wiley, 2009 — 320 с..

21. Laravel [Электронный ресурс] — Заглавие с экрана. Режим доступа: https://laravel.com/docs/9.x

22. Nixon R. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5 R. / Nixon — 5th ed. — O'Reilly Media, 2018. — 832 с.

23. PHP Data Objects [Электронный ресурс] — Заглавие с экрана. Режим доступа: https://www.php.net/manual/en/book.pdo.php

24. Stauffer M. Laravel: Up & Running: A Framework for Building Modern PHP Apps 2nd Edition / M. Stauffer — O'Reilly Media, 2019. — 554 с.

25. Tatroe K. Programming PHP: Creating Dynamic Web Pages / K. Tatroe, P. MacIntyre — 4th ed. — — O'Reilly Media, 2020 — 544 с.

26. Van Rijsbergen C.J. Information Retrieval, / C.J. Van Rijsbergen. — 2nd ed. — London: Butterworths, 1979. — 375 с.

27. Welling L. PHP and MySQL Web Development / L. Welling, L. Thomson — 5th ed. — Addison-Wesley, 2016. — 688 с.

28. W3Techs [Электронный ресурс] — Заглавие с экрана. Режим доступа: https://w3techs.com