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 «Разработка модуля для отображения событий СКУД на отдельных участках периметра»

Обучающийся В. Д. Логунов-Граф
(Инициалы Фамилия) (личная подпись)

Руководитель М. А. Тренина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

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

Тольятти 2022

## Аннотация

Тема бакалаврской работы: «Разработка модуля для отображения событий СКУД на отдельных участках периметра».

Актуальность работы заключается в том, что при просмотре базы данных, возникает необходимость фильтрации информации, поступающей при запросе.

Целью данной квалификационной работы является разработка модуля системы контроля и управления доступом (СКУД) для отображения событий на разных участках периметра. Разработанный в ходе работы программный продукт может помочь оптимизировать процесс поиска записей в базе данных СКУД.

В первом разделе рассматриваются различные виды алгоритмов поиска и канонические модели поиска информации.

Во втором разделе происходит программная реализация модуля, создание тестовой базы данных, имитирующей события СКУД, готовый модуль тестируется, а результаты его работы анализируются.

Выпускная квалификационная работа состоит из пояснительной записки объёмом в 42 стр. и содержит 23 рисунка, введение, два раздела, заключение и список используемой литературы, который включает в себя 25 источников.

# ABSTRACT

The title of the graduation work is «Development of a module for displaying ACS events in separate sections of the perimeter».

The senior paper consists of an introduction, two parts, a conclusion, list of references including foreign sources.

The key issue of the thesis is the development of a module for an access control and management system that allows you to receive and display events from separate sections of the perimeter.

The aim of the work is the software implementation of the ACS module, which will allow you to search for events received from a certain section of the perimeter in the database.

The graduation work may be divided into several logically connected parts which are: analysis of existing search models, development of the visual part and the main structure of the web application, creating a database that simulates the events of the access control and management system, software implementation of the search algorithm, module testing.

Finally, we received a fully operational ACS module that meets the requirements and performs the tasks correctly.

In conclusion we'd like to stress that this work is relevant not only for the implementation of the module of the access control and management system, but also for solving problems with similar requirements.

## Содержание

# Введение

На сегодняшний день в нашем мире все чаще приходится работать с большими объемами информации, для хранения которой необходимо иметь настолько же большой резерв носителей общего назначения или специализированных. При этом следует обратить внимание на то, что с увеличением объема информации становиться сложнее ее обрабатывать.

Для упрощения таких задач были созданы базы данных, которые позволяли хранить достаточно большие количества записей любого содержания в пределах допустимых форматов полей. Но даже с созданием баз данных следует учитывать, что поиск стал проще лишь для людей, обладающих определенными знаниями необходимыми для формирования запросов на языках программирования, что также доставляет неудобство при использовании для обычных пользователей базы данных.

Было найдено решение проблемы, заменяющее собой сложный или же затратный по времени процесс обработки записей базы данных, позволяющий использовать необходимые функции с любого устройства при этом с интуитивно понятным интерфейсом взаимодействия с пользователем. Это решение веб-приложение, загруженное на сервер с базой данных.

Целью данной работы будет являться разработка проекта, взаимодействующего с базой данных и выдающего записи, удовлетворяющие запросу пользователя.

Объектом исследования является изучение математических моделей поиска и составление алгоритма для их применения на базе данных.

Для достижения поставленной цели в работе решаются следующие задачи:

– анализ существующих математических моделей и алгоритмов поиска информации,

– изучение методов взаимодействия с базами данных,

– рассмотрение способов реализации WEB-приложений и основные инструменты WEB-разработки,

– тестирование и отладка программного модуля.

Данная работа содержит в себе введение, два раздела, заключение и список используемой литературы.

В первом разделе проходит рассмотрение и выбор из существующих математических моделей поиска, составляются блох-схемы.

Во втором разделе описывается разработка визуальной части, тестовой базы данных, которая будет имитировать события СКУД, позволяя протестировать веб-приложение, алгоритма поиска, также полученные блоки собираются в единый модуль, который проходит тестирование.

# 1 Описание алгоритма поиска

Результатом данной работы должно являться создание модуля для СКУД, который облегчит поиск записей, поступающих со входов и иных типов проходных в разные корпуса университета, удовлетворяющих условиям, заданным ЛПР (лицом, принимающим решения) о том, на каком участке периметра произошло событие. Для достижения поставленной задачи необходимо выполнить следующие пункты.

- Провести анализ и изучить расположение базы данных и данных в ней, для дальнейшей их обработки.
- Исследовать предметную область.
- Реализовать программный подход для решение поставленной задачи.
- Протестировать и отладить программный продукт перед введением для возможности введения его в эксплуатацию.

## 1.1 Математические модели поиска

В конце двадцатого века из-за быстрого развития технологий высокого уровня в сфере передачи и обработки информации, в том числе и создание систем телекоммуникаций, используемых даже в двадцать первом веке, начали появляться новые возможности организации многих этапов научно-информационного процесса, а это дало толчок для бурного роста информационных потребностей работников научных сфер. На данный момент одним из наиболее перспективных направлений развития информационного обеспечения научной деятельности являются информационные технологии в электронике. Мы затронем лишь те способы удовлетворения информационных потребностей, которые базируются на информационных технологиях в электронике.

Приступим к анализу математических моделей существующих алгоритмов поиска и ранжирования [8][10].

Состояние любой современной информационно-поисковой системы, в большинстве случаев определяют разработчики. Любой математический аппарат используется на уровне операций над различными множествами, векторной алгебры или теорией вероятностей. Именно так можно объяснить, то что обычно предлагаемые усовершенствования любых алгоритмов сводится к лингвистическим дополнениям. Модели поиска обычно делят на булеву, векторно-пространственную и вероятностную. Чтобы построить модель поиска чаще всего используют такой набор понятий:

– Множество документов, к которому применяется алгоритм { }

– Множество термов { ( ) }= ( ) , где ( ) – набор терминов (слов), содержащихся в документе к которому применяется алгоритм.

Рассмотрим булеву модель, которая основывается на теории множеств и математической логике. Представим документ и запрос как множество слов (термов). Терм будет представлять из себя булеву переменную такую, что она может равнять только 0 (в документе не найдено терма из запроса) и 1 (в документе найдем терм из запроса). У терма в документе есть весовое значение, которое принимает только два значения $w^{(i,j)} \in \{0,1\}$.

Особенность булевой модели поиска в том, что пользователь в ней имеет возможность сформулировать свои запросы как булево выражение, применяя при этом операторы И, ИЛИ, НЕТ. Исходя из факто о том, что каждое логическое выражение можно представить, как дизъюнкцию некоторых выражений, которые соединены между собой операциями конъюнкции, или же как ДНФ (дизъюнктивная нормальная формула). Можем записать в виде формулы (1):

$$q \equiv d_{dnf} \equiv V_{i=1\ldots N} q_{cc}^{(i)} \quad (1)$$

где

– q – инверсная функция;

– $q_{cc}^{(i)}$ – $i$-й конъюнктивный компонент запроса $q_{dnf}$.

Мера близости документа $d^{(j)}$ и запроса q – sim($d^{(j)}, q$) для булевой модели будет определяться через формулу (2):

$$sim(d^{(j)}, q) = \begin{Bmatrix} 1, \text{если } \exists q_{cc}^{(i)} : (q_{cc}^{(i)} \in q_{dnf}) \wedge (\forall k, g_k(q_{cc}^{(i)}) = g_k(d^{(j)}) \\ 0, \text{иначе} \end{Bmatrix} \quad (2)$$

где

- $g_k$ – инверсная функция, которая соответствует индексу терма $t_k$, определяющаяся как $g_k(d^{(j)}) = w_k^j$.

Таким образом sim( $d^{(j)}, q$ ) = 1, в том случае, когда существует конъюнктивная компонента $q_{cc}^{(i)}$ , которая входит в дизъюнктивную нормальную форму $q_{dnf}$, такая что инверсная функция любого из термов к данной конъюнктивной компоненте совпадет с этой же инверсной функцией для документа $d^{(j)}$. В том случае, если это не так, тогда sim($d^{(j)}, q$) = 0.

Исходя из этого, когда sim($d^{(j)}, q$) = 1, тогда согласно булевой модели документ $d^{(j)}$ считается востребованным для запроса q, а иначе, если sim($d^{(j)}, q$) = 0, то $d^{(j)}$ будет считаться невостребованным по запросу q.

Главным преимуществом булевой модели поиска является то, как просто ее реализовать. Но есть и недостатки, такие как:

– Невысокий показатель успешности поиска, нет возможности ранжировать полученные результаты по востребованности, из-за отсутствия критериев.

– Важность наличия у пользователя навыков и умения, необходимых для правильного формирования запросов с применением булевых операторов.

Существенный недостаток классической булевой модели основан на том, что в ней нет значений весов для термов при запросе, что приводит к пренебрежению значимостью некоторых термов. Из-за этого становиться

невозможным сортировка для результатов поиска по тому насколько они соответствуют изначальному информационному запросу. Для устранения недостатков и полноценного использования преимуществ были выведены некоторые расширенные булевы модели. В них появляются специальные обобщения операторов, основываясь на нечетких множествах, что делает возможным вести учет меры соответствия запросу.

Теперь рассмотрим векторную модель поиска. Она является типичным представителем алгебраических моделей. Для этой модели каждый документ и запрос представлен как вектор в многомерном пространстве термов. Всякому из термов, встречающихся в документе, сопоставляется значение его веса. Весовое значение определяется на основании статистической информации о частоте нахождения терма в документе, который рассматривается, а также, во всех документах, найденных по запросу. В данной модели нет возможности использовать любые логические операции при составлении запроса. При оценке релевантности документа для данного запроса используют скалярное произведение для вектора «запрос» и вектора «документ».

Уровень соответствия документа $d^{(j)}$ к запросу $q$ представим в виде скалярного произведения векторов с информацией, имеющих весовые значения термов $\overline{d^{(j)}} = (w_1^{(j)}, w_2^{(j)}, \cdots, w_n^{(j)})$ и $\bar{q} = (w_1^q, w_2^q, \cdots, w_n^q)$. Вес любого из других термов вычисляют разными по-разному. Одним из простых методов основывается на том, что у веса терма $w_i^{(j)}$ используют нормализованную частоту $freq_i^{(j)}$ частоты нахождения для данного документа, учитывая его частоту встречаемости для других документов из списка. Способ называется учетом дискриминационной силы терма (3):

$$w_i^{(j)} = freq_i^{(j)} \times log \frac{N}{n_i}\ , \qquad (3)$$

где

– $n_i$ – число файлов, внутри которых используется терм $t_i$,

– $N$ – численность всех документов в множестве.

Исходя их этого можно привести пример, что в том случае, когда слово встречается во всех файлах ($n_i = N$), то расчет формулы (3) даст в результате 0, из чего следует бесполезность запроса.

Данный метод взвешивания термов обозначается как TF*IDF (от англ. Term Frequency – частота термина * от англ. Inverse Document Frequency – обратная частота документа), что обозначает произведение частоты встречаемости терма в документе на число, обратное количеству документов в массиве, содержащих данный терм.

При вычислении тематической близости документа к запросу, в данной модели применяются простые скалярные произведения $\mathrm{sim}(d_j, q)$, которые соответствуют косинусу угла между двумя векторами документа и запроса. Уровень соответствия документа $d^{(j)}$ для запроса $q$ будет являться вычисленная по формуле (4) величина:

$$sim(d_j, q) = \frac{\overline{d^{(j)}}}{|d^{(j)}|} \times \frac{\bar{q}}{|\bar{q}|} = \frac{\sum_{i=1}^{n} w^{(j)} w_i^q}{\sqrt{\sum_{i=1}^{n} \left(w_i^{(j)}\right)^2} \times \sqrt{\sum_{i=1}^{n} \left(w_i^q\right)^2}} \ . \qquad (4)$$

где

– $d^{(j)}$- документ для запроса,

– $q$ – запрос,

– $w$ – элементы вектора запроса или документа.

Применение модели поиска, основанной на векторах, встречается гораздо чаще остальных. Все дело в достаточно простой реализации и обеспечивает высокую продуктивность поиска с упорядочиванием результатов путем, ранжирования. Помимо всего векторно-пространственная модель создает возможность для упрощенного поиска подобных документов. Каждый документ будет рассмотрен как запрос. Основным недостатком векторной модели поиска считается высокая затратность ресурсов при

обработках массивов большая размерности, из-за чего в каноническом виде непригодна для работы с интенсивной нагрузкой.

В основе вероятностной модели поиска лежит теория вероятностей. востребованность для этой модели представлена в виде вероятности такого исхода, при котором документ заинтересует пользователя. Для правильности работы алгоритма, необходимо, чтобы существовал набор файлов, удовлетворяющих пользователя (учебная выборка), которые он сам выбрал или же те, которые предоставлены при каком-либо упрощенном предположении. Для расчета вероятности удовлетворения пользователя, у каждого из множества документов вычисляется соотношение того, как часто терм встречается в множестве, удовлетворяющем пользователя и его встречаемостью в неудовлетворяющей части того же множества.

Основная вычислительная нагрузка данной модели заключается в вычислении вероятности, того, что документ удовлетворяет пользователя, путем предположения того, что термы из запроса распределены как среди удовлетворяющих файлов, так и внутри неудовлетворяющих файлов. Для вычисления этих вероятностей используются формулы, которые основываются на теореме Байеса [20], в соответствии с которой, по некоторой функции вероятностей получим конечную форму, которая оценивает уровень вероятности удовлетворения для каждого документа из учебной выборки, называемую поисковым статусом формулу (5)

$$SV = \sum_{t_i \in q \cap d} SV = \sum_{t_i \in q \cap d} log \frac{rel_i(nrel - nrel_i)}{nrel_i(rel - rel_i)}, \quad (5)$$

где

- $rel_i$ – количество документов удовлетворяющих пользователя и содержащих i-ый терм,
- $nrel_i$ – количество документов неудовлетворяющих условию, d – учебная выборка документа рассматривается, как множество слов,

– q – множество слов содержащихся в запросе, $q \cap d$ – пересечения множеств, дающее новое множество содержащее в себе термы присутствующие как в запросе, так и в документе.

Модели поиска, основанные на вероятности, имеют теоретическое превосходство, за счет наиболее правдоподобного способа формального описания проблемы информационного поиска и при наличие исходных данных могут показать отличное прогнозирование востребованности.

Проанализировав все алгоритмы, описанные выше, можно считать, что любая модель в своей канонической форме имеет недостатки. Таким образом вероятностная модель неудобна из-за необходимости разработки алгоритмов для обучения и малой масштабируемости. Векторная модель имеет возможность высокоточной работы с большими объемами данных лишь за счет серьезного аппаратного оснащения. Булева модель нуждается в наличие хотя бы минимальных навыков и знаний логической алгебры, так же она ограничена четким набором операторов, но главный недостаток в отсутствии возможности упорядочивания результатов по их релевантности.

На данный момент для успешности использования моделей поиска, используются гибридные подходы. Примером такого подхода может являться такой алгоритм, где результаты находит булева модель, а ранжированием результатов векторная, что позволяет использовать лишь положительные качества каждой из моделей. Еще один способ применения, это использование расширенные модели, для которых характерно использование дополнительных алгоритмов по анализу запросов, а точнее определение наличия взаимосвязи между термами в запросе или лингвистический анализ так далее. Можно сделать вывод, что даже при эффективном распределении нагрузки на различные блоки информационного поиска, его затратность ресурсов все так же остается значительной, что приводит к необходимости сборки более мощной аппаратной части и как следствие к увеличению стоимости всей системы в целом. Из-за этого при построении математической модели стоит предоставить немало внимания

выбору таких параметров поиска и оценки востребованности пользователем, которых, при минимальном количестве, должно хватать для показателей работоспособности алгоритма не хуже, чем у существующих алгоритмов.

### 1.2 Разработка алгоритма поиска по запросу

Для того, чтобы программный продукт можно было считать работоспособным, удобным и оптимизированным, следует разработать несколько отдельных страниц, каждая из которых будет иметь собственный функционал. Для каждой из этих страниц необходимо создать возможность перемещения на другие страницы приложения.

Граф, показывающий возможность перемещения между страницами проекта (рисунок 1)

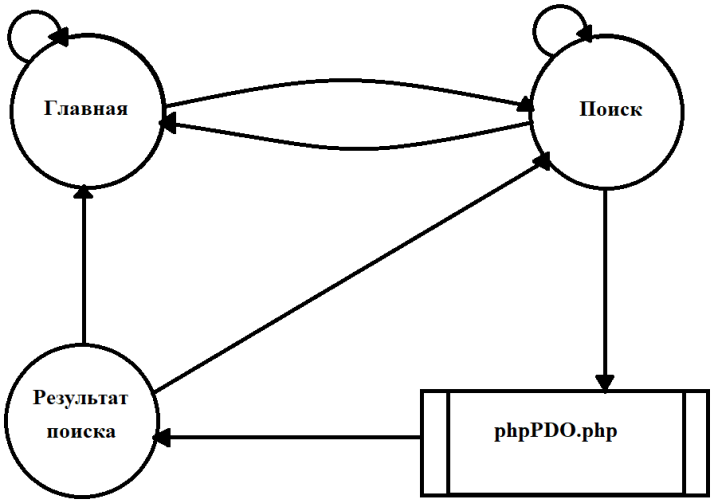


Рисунок 1 - Граф возможных перемещений между страницами проекта

Рассмотрим подпрограмму phpPDO.php.Это блок программного кода на языке PHP, выполняющий основную часть вычислений для модуля СКУД. Блок-схема [2] phpPDP.php (рисунок 2).

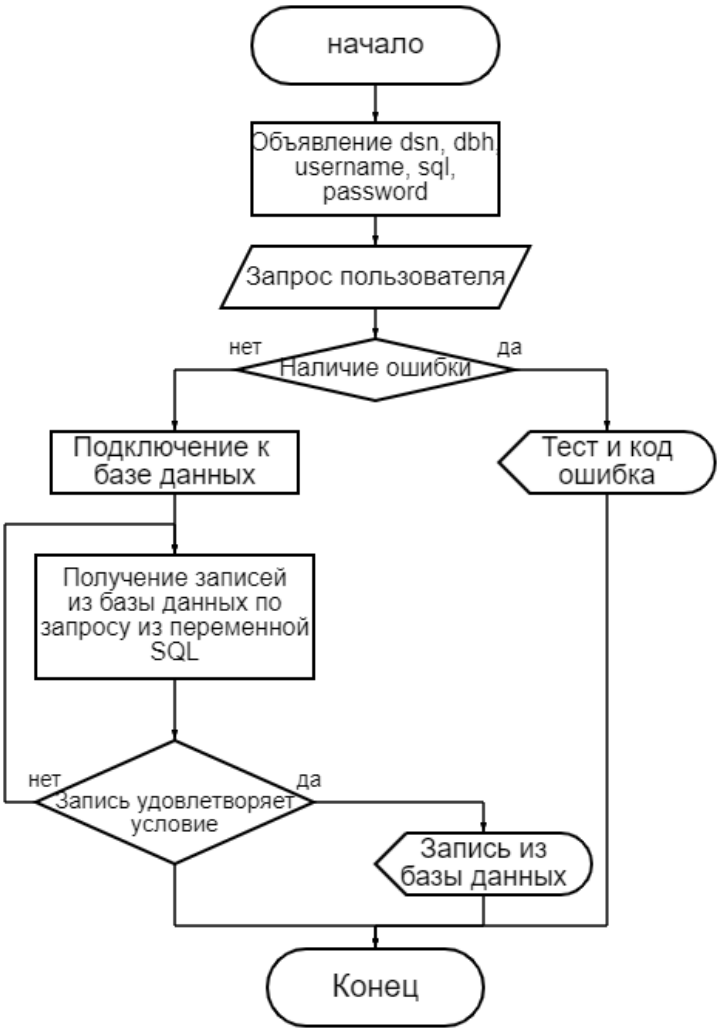


Рисунок 2 - Блок-схема алгоритма поиска

Так выглядит схема алгоритма, который выполняет поиск по запросу среди записей базы данных. Согласное данным, указанным на рисунке 2, будет составлен программный код phpPDO.php.

Вывод по первому разделу:

В первом разделе происходит изучение предметной области.

Происходит постановка задач, приводятся и анализируются основные математические модели поиска. Также показываются блок-схема программной реализации алгоритма поиска записей и граф, позволяющий наглядно увидеть возможности перемещения между страницами проекта.

## 2 Программная реализация модуля СКУД

В качестве модуля СКУД (системы контроля и управления доступом) будет разработано веб-приложение [5][7][18], которое позволит взаимодействовать с базой данных при помощи любого устройства в локальной сети, а так же выполнять запрос для получения записей с определенных участков периметра, намного проще, чем при использовании инструментов, поставляемых с СУБД.

Процесс создания будет разделен на три основных этапа:

– Разработка части модуля, которая будет предоставлена для взаимодействия с пользователем при открытии веб-приложения.

– Создание тестовой базы данных, которая позволит убедиться в работоспособности продукта, за счет имитации записей системы контроля и управления доступом.

– Выбор инструментов для реализации алгоритма поиска, с дальнейшим написанием блока, отвечающего за поиск записей по условию.

По завершению разработки всех блоков программной части, они будут объединены в веб-приложение.

### 2.1 Разработка визуальной части веб-приложения

Визуальной частью модуля будет называться все, что относится к стандартному представлению о веб-приложении, то есть программная реализация, страниц, контроллеров, форм, кнопок и всего, что будет взаимодействовать с пользователем на уровне интерфейса.

Для разработки визуальной части веб-приложения будет использоваться бесплатный фреймворк с открытым кодом Laravel [3][23], чтобы создать необходимую архитектуру модели MVC и использовать инструменты нужные для правильной работы проекта. Выбор фреймворка можно обосновать его распространенностью среди веб-разработчиков, что

делает его изучение проще, за счет активного распространения материалов с различной информацией, имеющей пользу для неопытных разработчиков. Получить веб-фреймворк Laravel можно при помощи бесплатного пакетного менеджера Composer путем введения команд установки в терминал проекта или же загрузив его с официального сайта разработчика. После установки нужных средств для построения архитектуры, приступим к созданию основы проекта. При установке веб-фреймворка в указанную директорию, в ней появятся все необходимые для начала работы компоненты веб-приложения. Схематично представим те директории, которые нам необходимы и будут использованы в ходе разработки (рисунок 3).

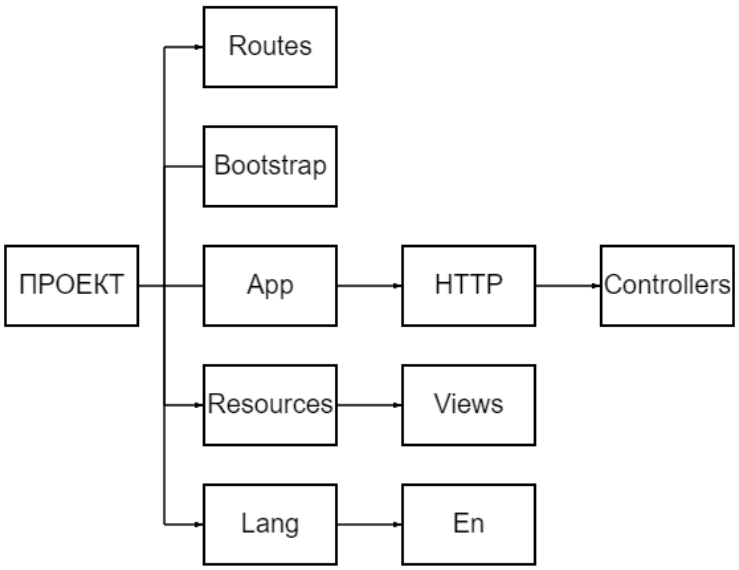


Рисунок 3 – Схематичное представление директорий, используемых в ходе разработки проекта

Также, помимо директорий, указанных на схеме, предоставляются различные файлы конфигураций, интерпретаторы кода программ, написанных на некоторых языках программирования, со стороны сервера и другие инструменты, которые помогают в создании и использовании веб-приложения.

Для проверки правильности работы всех служб проекта запустим его на локальном сервере при помощи artisan, затем переходим на локальный сервер по адресу, указанному при запуске.

Также для расширения возможностей в области дизайна воспользуемся фреймворком Bootstrap [25], который позволит улучшить визуальную часть проекта и упростить ее написание. Выбор можно обосновать наличием возможности отображения одного и того же оформления на разных устройствах, простотой в использовании, а также широким выбором инструментов и готовых решений.

Следующий шаг при разработке визуальной части веб-приложения это создание веб-страниц и их интерфейсов, которые будут использоваться. Так как создание идентичной разметки и дизайна для каждой страницы не является оптимальным по времени и усилиям подходом при реализации оформления, то будем использовать однократное создание веб-страницы, содержащей внутри себя шаблон для создания остальных, включающий в себя все блоки, которые используются во всех веб-страницах и те, которые будут принимать в себя различную информацию, индивидуальную для большинства веб-страниц.

Создаваемый шаблон для оформления назовем layout и создадим соответствующий файл внутри проекта. Для этого заходим в директорию, в которой хранится проект, затем в каталог, который находиться в «'Название проекта'/app/resources/views» и создаем файл «layout.blade.php”, таким образом файл должен иметь путь «'Название проекта' /app /resources /views /layout.blade.php»

Внутри данного файла создадим все нужные блоки готового дизайна и разметки, а также те, которые позволят пользователю видеть информацию индивидуальную для каждой страницы, указывать свой заголовок, а также свое содержание «тела» страницы, а еще предоставит возможность свободно перемещаться по веб-приложению за счет добавления кнопок, позволяющих переходить на разные веб-страницы проекта.

Для того, чтобы приложение обладало требуемыми для удобства функциями, описанными ранее, будут использованы основные теги языка гипертекстовой разметки HTML [4][13] и метод «@yield», указывающего на вставку материала при вызове функции «@extends('layout') и @section(), @endsection с названием блока в который будет вставляться информация. В данном случае будут использованы блоки 'Title' для указания заголовка страницы и 'Body' для вставки материалов страницы.

Подробный код, написанный для оформления и работоспособности страницы layout.blade.php (рисунок 4).

diplom > resources > views > layout.blade.php

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@yield("Title")</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">
    <style type='text/css'>
            A { text-decoration: none;}
            A: hover {text-decoration: underline;
            color: dark;}
    </style>
</head>
<body class="bg-white">
    <div class="d-flex flex-column flex-md-row align-items-center p-3 px-md-4 mb-3 bg-white border-bottom shadow-sm">
    <h3 class="my-0 mr-md-auto font-weight-normal">TSU</h3>
    <nav class="my-2 my-md-0 mr-md-3">
        <a class="p-2 text-dark" href="/">Главная</a>
        <a class="p-2 text-dark" href="/search">Поиск</a>
    </nav>
    </div>
<div class="container">@yield('Body')</div>
</body>
</html>
```

Рисунок 4 – Код «layout.blade.php»

В том же каталоге, где располагается layout.blade.php, создадим еще два файла и назовем их home.blade.php и search.blade.php.

Файл «home.blade.php» будет являться главной страницей, предназначенной для перемещения по каталогам. Эта веб-страница создана для возможности присоединения дополнительных модулей в случае необходимости доработки или усовершенствования модуля СКУД при необходимости. Для создания home будет использована ранее описанный

шаблон, в которую мы помещаем на место заголовка страницы «Главная», а в блок содержания не передаем информацию, так как мы используем главную страницу лишь для перемещения по веб-приложению и наличия перспективы расширения возможностей и функционала.

В файле «search.blade.php» будет располагаться форма, участвующая в передаче данных в переменную необходимую для осуществления поиска записей в базе данных. Создание веб-страницы будет происходить при помощи шаблона, в которой мы передаем значения заголовка «Поиск», а в блок с содержанием будет помещена сама форма для ввода текста и кнопка для того, чтобы отправить введенные данные на обработку и валидацию. Так же на этой странице будет выводиться список ошибок при неверном заполнении формы. Для того, чтобы ошибка, выводимая при недопустимом содержании формы, была понятна пользователю, необходимо перейти в директорию «Название проекта \lang \ en» открыть файл «validation.php», найти строку с параметром «’required’» и заменить содержание выводимого предупреждения на собственное.

Подробный код, написанный для страниц home.blade.php, search.blade.php, будет изложен ниже (рисунок 5) (рисунок 6).

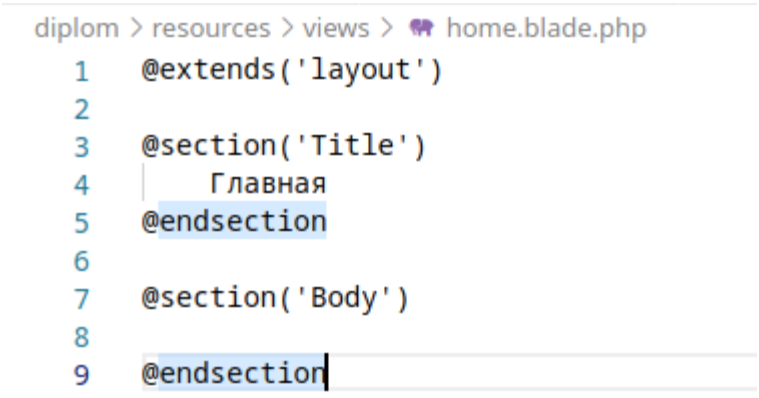


Рисунок 5 – Код «home.blade.php»

diplom > resources > views > search.blade.php

```
@extends('layout')

@section('Title')
    Поиск
@endsection

@section('Body')

@if($errors->any())
    <div class="alert alert-danger">
        <ul>
            @foreach($errors->all() as $error)
                <li>{{ $error }}</li>
            @endforeach
        </ul>
    </div>
@endif

<form method='post' action='/search/result'>
    @csrf
    <input type="corp" name="corp" id="corp" placeholder="Введите корпус для поиска" class="for-control">
    <button type="submit" class="btn btn-success">Найти</button>
</form>
@endsection
```

Рисунок 6 – Код «search.blade.php»

Если скомпилировать код, описанных выше файлов, то, в качестве результата, можно получить веб-страницы, которые позже будут предоставлены для взаимодействия с пользователем при входе и процессе поиска. Стоит учитывать, что элементы языка разметки HTML5 могут по-разному отображаться в браузерах, но они останутся работоспособными и несильно сменят оформление.

Результаты компиляции кода страниц (рисунок 7) и (рисунок 8).


Рисунок 7 - Страница «home»

Теперь, когда веб-страницы готовы, приступим к созданию класса, отвечающего за исполнение алгоритмов действий, которые будут выполняться при переходе на определенные страницы модуля.


Рисунок 8 - Страница «search»

Создадим контроллер [11] при помощи интерфейса командной строки artisan и назовем его MainController.php. Внутри него будут созданы функции для каждой из веб-страниц, которые мы будем использовать в нашем веб-приложении.

Сопоставим описанные функции контроллера с соответствующими им веб-страницами таким образом, что всякий раз при вызове метода home и search нам будут возвращаться и отображаться описанные ранее веб-страницы home.blade.php и search.blade.php соответственно.

Полученный код контроллера (рисунок 9).

diplom > app > Http > Controllers > MainController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class MainController extends Controller
{
    public function home(){
        return view('home');
    }
    public function search(){
        return view('search');
    }
    public function search_result(Request $request){
        $valid = $request->validate(['corp'=> 'required|min:1|max:100']);
        echo '<head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Результат поиска</title>
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">
        <style type="text/css">
                A { text-decoration: none;}
                A: hover {text-decoration: underline;
                color: dark;}
        </style>
    </head>
    <body class="bg-white">
        <div class="d-flex flex-column flex-md-row align-items-center p-3 px-md-4 mb-3 bg-white border-bottom shadow-sm">
        <h3 class="my-0 mr-md-auto font-weight-normal">TSU</h3>
        <nav class="my-2 my-md-0 mr-md-3">
            <a class="p-2 text-dark" href="/">Главная</a>
            <a class="p-2 text-dark" href="/search">Поиск</a>
        </nav>
        </div>';
        echo '</body> </html>';
    }
}
```

Рисунок 9 – Код контроллера «MainController.php»

За перемещение по страницам проекта отвечает файл web.php, который при переходе на определенный адрес будет запускать соответствующие методы класса MainController, поэтому при переходе «'адрес сервера'/» будет выполнен метод home типа get и вернет страницу home.blade.php, а «'адрес сервера'/search» будет выполнен метод search типа get и вернет страницу search.blade.php. Так же создадим один post функцию при переходе «'адрес сервера'/search_result», которая будет отправлять в метод MainController search_result данные из формы на валидацию и для составления запроса для базы данных.

Весь код, описанный внутри файла web.php (рисунок 10).

diplom > routes > web.php

```php
<?php

use     App\Http\Controllers\MainController;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', [MainController::class,'home']);

Route::get('/search', [MainController::class,'search']);

Route::post('/search/result', [MainController::class,'search_result']);
```

Рисунок 10 – Код «web.php»

Интерфейс для взаимодействия с пользователем, будущего модуля системы контроля и управления доступом, готов. После его реализации стало возможно увидеть визуальную часть проекта, а также переходить на различные страницы веб-приложения.

## 2.2 Создание базы данных для тестирования модуля

Согласно требованиям, описанным в постановке задачи, итоговый продукт должен быть предназначен для использования совместно с СУБД Firebird [6][16], именно поэтому вся реализация будет на примере баз данных этого типа.

Преимуществами выбранных баз данных будет являться то, что она поддерживается на многих операционных системах, это бесплатный программный продукт, имеет высокую эффективность, поддерживает большое количество языков для процедур и триггеров и самым главным можно считать возможность поддерживать параллельные обработки оперативных и аналитических запросов, за счет того, что читающий пользователь не блокирует пишущего. За счет всех этих преимуществ, именно эту систему управления базами данных следует использовать при оснащении и составлении системы контроля и управления доступом.

Для тестирования работоспособности нашего проекта создадим базу данных [17][19][21], которая будет использоваться в ходе проверки программного продукта на соответствие поставленным требованиям.

При создании укажем, владельцем пользователя «SYSDBA», а его пароль зададим как «123», каталог для хранения создаваемой базы данных и ее название выберем произвольно.

Используя данные владельца, заданные ранее, подключимся к созданной базе данных как пользователь SYSDBA и создадим в ней таблицы с названиями «EMPLOYEES» и «EVENTS». Первая таблица содержит в себе автоинкрементное, ключевое, обязательное для заполнения поле «ID» целочисленного типа, обязательное для заполнения полем «NAME» символьного типа с максимальной размерностью 60, эти поля будут использованы для хранения идентификатора объекта, его имени и отчества. Вторая таблица включает в себя ключевое, ненулевое целочисленное поле

«ID» и обязательное для заполнения поле «CORP» символьного типа, с размерность 10 символов.

Свойства таблиц «EMPLOYEES» и «EVENTS» (рисунок 11) (рисунок 12). Схема базы данных (рисунок 13).

Чтобы получить максимальную точность у разрабатываемого модуля, необходимо проводить тестирование алгоритма при наиболее похожих условия, для этого при помощи стандартных команд редактирования группы CRUD (Create Read Update Delete), сымитируем данные хранящиеся и получаемы в ходе работы системы контроля и управления доступом. Создадим записи о 10 сотрудниках, в таблице «EMPLOYEES» нашей базы данных и 10 событий для этих пользователей в таблице «EVENTS» и выведем на экран все заполненные поля этих записей, для проверки правильности содержания полей информацией. Установим такое условное ограничения по заполнению поля CORP, что туда будут вводиться корпусы ТГУ, так как мы заполняем тестовую базу данных, то имена будут взяты произвольно, а корпус выбран случайно.

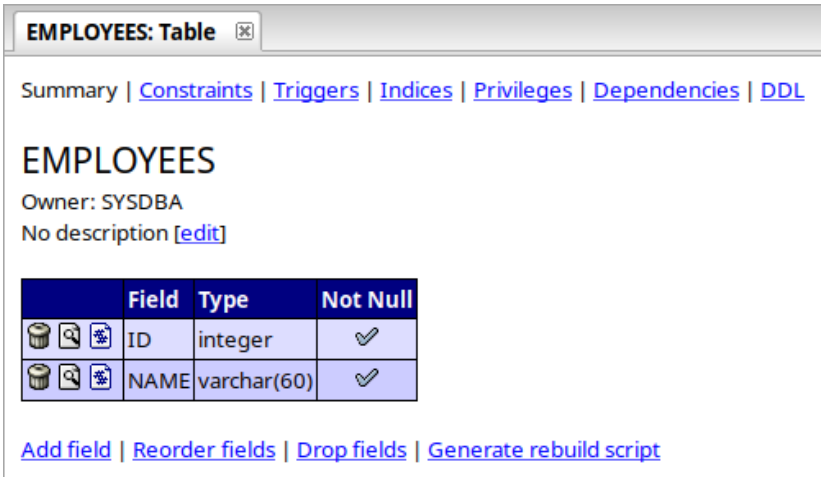


Рисунок 11 – Свойства таблицы «EMPLOYEES»

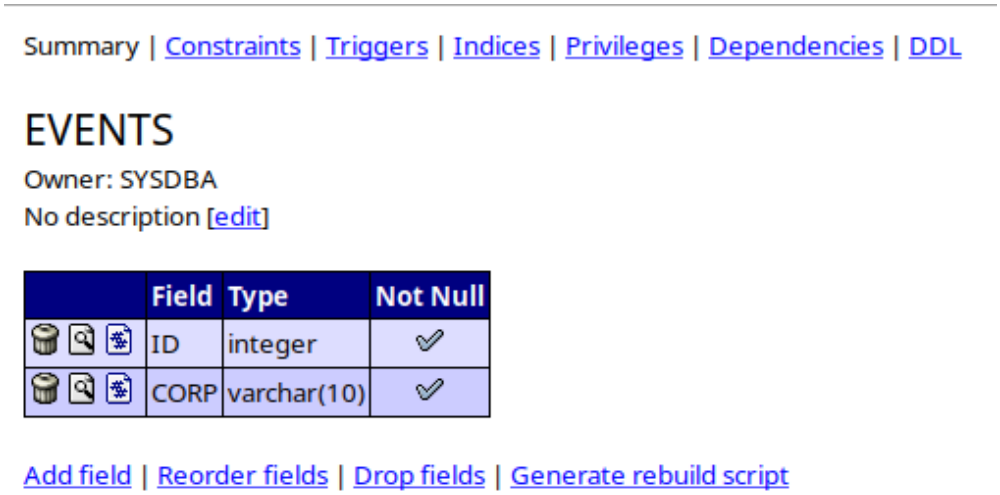


Рисунок 12 – Свойства таблицы «EVENTS»

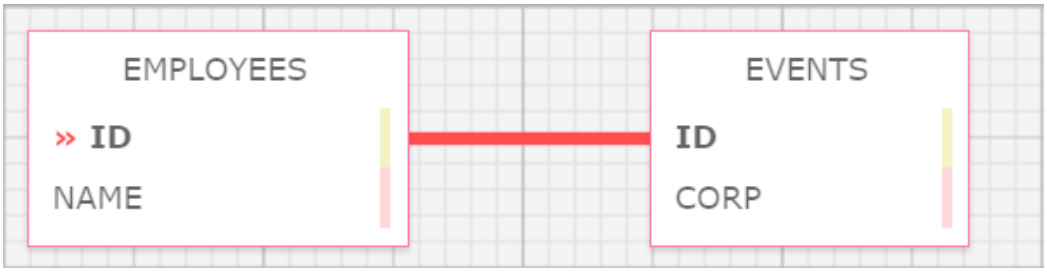


Рисунок 13 – Схема базы данных «TSU»

Вывод всех данных из таблицы «EMPLOYEES» (рисунок 14).

```
SQL> SELECT * FROM EMPLOYEES;

          ID NAME
============ ============================================================
           1 Иван Иванов
           2 Олег Николаевич
           3 Виктор Семёнович
           4 Кирилл Игнатьевич
           5 Трофим Ефимович
           6 Максим Дмитриевич
           7 Ульяна Викторовна
           8 Анна Петровна
           9 Мария Михайловна
          10 Матильда Ильинична

SQL>
```

Рисунок 14 - Вывод всей информации из таблицы «EMPLOYEES»

Результат запроса всех записей из таблицы «EVENTS» (рисунок 15).

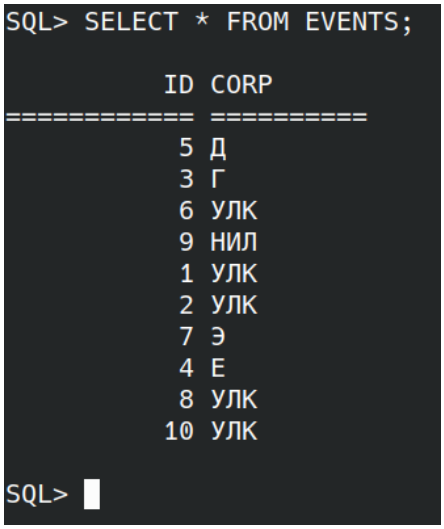


Рисунок 15 - Вывод всей информации из таблицы «EVENTS»

Полученные данные будут использоваться при тестировании модуля, для определения правильности результатов работы приложения.

### 2.3 Разработка алгоритма для поиска записей в базе данных

При выборе языка программирования для разработки программной части поиска будут рассмотрены PHP, JS и Scala.

Язык PHP отвечает за выполнение кода на стороне веб-сервера. Он имеет достаточную мощность и гибкость, из-за чего и стал настолько популярным среди веб-разработчиков. PHP, согласно статистике, используется при разработке около 60% веб-приложений. Данный язык программирования взаимодействует с базами данных при помощи расширения «Firebird/Interbase» или драйвера PHP PDO.

Преимущества этого языка в том, что он предоставляется бесплатно, достаточно прост при освоении, за счет влияния на него многих распространенных языков программирования, таких как C++, Java, JavaScript, HTML, также имеет широкий функционал при операциях с базами данных

[12]. Из недостатков стоит отметить, что данный язык имеет проблемы с безопасностью и объекты передаются не при помощи ссылок, из-за чего этот язык не подходит для разработки крупных интернет ресурсов.

Язык программирования R. Прямой потомок старшего языка S, поддерживается Фондом языка R для статистических вычислений (R Foundation for Statistical Computing). К его преимуществам можно отнести наличие большого количества пакетов для выполнения различных задач, при предоставлении, бесплатно поставляет с базовой версией большое количество встроенных функций и методов, а также имеет возможность для графического отображения данных. Недостатки у этого языка программирования тоже существуют. Низкая производительность, слишком сильная предрасположенность для решения нестандартных и специфичных задач, имеет особенности, которые вызывают неудобства при использовании R опираясь на опыт в других языках. Перечисленные качества делают R языком отличным для решения нестандартных задач с широким набором расширений 2и методов, но неудачным решением задач, связанных с высокой вычислительной нагрузкой.

Рассмотрим JavaScript. Язык является одним из лидеров по популярности, так же он существует достаточно долго и смог стать мощным инструментом, применяемым в IT-сфере. К его достоинствам стоит отнести достойную инфраструктуру, производительность и скорость работы, а также легкость в освоении. Стоит отметить, что JavaScript, также, как и любой другой язык программирования, имеет ряд недостатков, например его отсутствие возможности работы с файлами, достаточно слабая устойчивость перед злоумышленниками и отсутствие возможности использования для сетевых приложений.

Бесплатный язык программирования Scala, основанный на базе JVM. Поддерживает использование как объективно-ориентированный подход, так и функциональный. Из преимуществ можно отметить его высокую производительность и возможность работы с большими объемами данных и

компиляция байт-кода Java, что позволяет ему хорошо поддерживать этот язык. Главным недостатком является сложность в его изучении и применении.

Таким образом можно выделить четыре языка для работы с базами данных. Названия, преимущества и недостатки отобранных языков программирования (рисунок 16).

Проанализировав все возможности, достоинства и недостатки языков программирования PHP, JavaScript и Scala, было принято решение использовать язык PHP, за счет его предрасположенности к работе с файлами и базам данных, что делает его лучшим выбором для реализации решения данной задачи, а его недостатки, такие как проблемы с безопасностью и несоответствие уровню языка для разработки крупных веб-ресурсов, достаточно незначительными.


Рисунок 16 – Варианты языков программирования для программной реализации алгоритма поиска записей в базе данных

Для реализации части программы, отвечающей за поиск, выбираем скриптовый язык общего назначения PHP из-за его интенсивного применения в разработке веб-приложений [15][24].

Изначально язык PHP может справиться лишь с некоторой частью, поставленной нами задачи, он может интерпретироваться браузером при исполнении языка разметки HTML и с его помощью можно составлять основные алгоритмы поиска, сортировки и обработки информации через переменные и их производные. Для расширения возможностей и функционала программного кода необходимо установить расширения, которые позволять использовать язык в новых сферах, нужных нам для осуществления поиска записей в базе данных.

Так как нам предстоит взаимодействие с системой управления базами данных Firebird, то нам необходимо установить соответствующее расширение, которое позволит подключаться к базам данных, а также правильно отображать и интерпретировать информацию, содержащуюся внутри них.

Для возможности манипуляций над базами данных и их содержимым так же необходимо устанавливать соответствующие расширения. У PHP есть несколько вариантов, подходящих для этой задачи, они называются Firebird/Interbase расширение и PHP PDO драйвер [23].

Firebird/Interbase является наиболее проверенным продуктом и использует процедурный подход к написанию программ. Основными инструментами этого расширения являются функции с префиксом ibase_, которые могут возвращать или принимать либо соединение, либо транзакцию, либо подготовленные запрос SQL или результат запроса группы CDUR. Именно так выглядит стиль программирования с применением данного метода для реализации манипуляций с базой данных

Следующий вариант, позволяющий работать с информацией внутри базами данных это драйвер PHP PDO. Его преимущество перед Firebird/Interbase в том, что он предоставляет обобщенный интерфейс для доступа к различным типам баз данных. Таким образом каждый тип системы управления базами данных, для которой установлено соответствующее расширение, может представить специфичный для базы данных функционал

в виде стандартных функций расширений. Стиль программирования PDO представляет из себя объектно-ориентированный подход в написании программ, иными словами, требует понимания объективно-ориентированного программирования [1][9], что позволяет существенно упростить работу с данными. За то какой именно драйвер будет использовать PDO, будет отвечать так называемая строка подключения DSN (Data Source Name), устройство этой строки позволяет определять такой параметр, как префикс, отвечающий за определение типа базы данных. Для каждого из типов базы данных так же описаны свои параметры подключения, которые описаны в документации PDO драйвера. При создании объекта PDO от его базового класса, соединения устанавливаются автоматически.

Исходя из соображений долгосрочности, будет выгоднее использовать метод с использованием драйвера PHP PDO [14][16][22], что позволяет сохранить продукт в случае миграции, путем замены пары параметров подключения и изменением синтаксиса некоторых запросов.

Опираясь на выбранный метод, теперь нам необходимо написать программный код, который будет осуществлять поиск записей, удовлетворяющих условию поиска, и выводить на странице результаты по запросу от пользователя.

Для начала опишем переменную DB, в которую, согласно документации, передаем такие параметры как тип базы данных, ее расположение и тип кодирования символов. Строка составленная на данном шаге имеет вид «'firebird:dbname=путь к базе данных; charset=тип кодировки;' ». Затем так же создадим переменные, содержащие внутри себя имя пользователя и его пароль. Исходя из данных указанных ранее при описании свойств тестовой базы данных, мы создадим переменную NAME и PASSWORD равными строкам «SYSDBA» и «123», соответственно. Далее для подключения мы создадим метод try в котором будет проходить само подключение и основной вычислительный процесс, а также создадим метод catch, выводящий любые ошибки как результат работы программы при их

появлении, что послужит удобным инструментом при возможной необходимости отладки продукта. Внутри try производим подключение путем составления переменной DSN из нового объекта класса PDO, в который передаем переменные DB, NAME, PASSWORD и свойства соединения, используя методы описанные в документации к драйверу PDO, в нашем случае передадим параметры, позволяющие передавать любые возникающие ошибки в метод catch для их отображения. Следующим шагом будет формирование запроса, он должен соответствовать синтаксису запросов Firebird и помещен в какую-либо переменную. Введем переменную с названием SQL и поместим туда строку с запросом, возвращающим все записи тестовой базы данных, созданной ранее, в которых корпус будет соответствовать строке, введенной с клавиатуры. Таким образом представим, что мы имеем некую переменную SEARCH, вводимую с клавиатуры, и запрос на поиск записей, удовлетворяющих условия в базе данных. Получаемая строка имеет вид «'SELECT * FROM TSU WHERE CORP='.”'”.SEARCH.”'”» и содержится внутри переменной SQL. Теперь осталось лишь применить заданный запрос к подключенной тестовой базе данных. Опишем переменную query такую, что внутри нее производится запрос SQL по переменной подключения к базе данных DSN и построчно выведем на печать все интересующие нас данные в виде объекта, задав это в виде параметра метода fetch. Для тестовой базы данных будут выводиться идентификаторы, имена и корпуса, поэтому указываем все существующие поля в цикле, чтобы через переменную $row они поместились в качестве объектов в очередь передачи данных на экран.

Для того, чтобы убедиться в правильности написанной программы, сформируем и выполним запрос на тестовой базе данных, выводящий только те записи, поля «CORP» которых содержат «УЛК». Для этого запустим программу и введем с клавиатуры в переменную «$SEARCH», отвечающую за формирование запроса строку, хранящую в себе название корпуса, для

которого будут выводиться записи о сотрудниках, закрепленных за ним. Результат работы программы (рисунок 17).

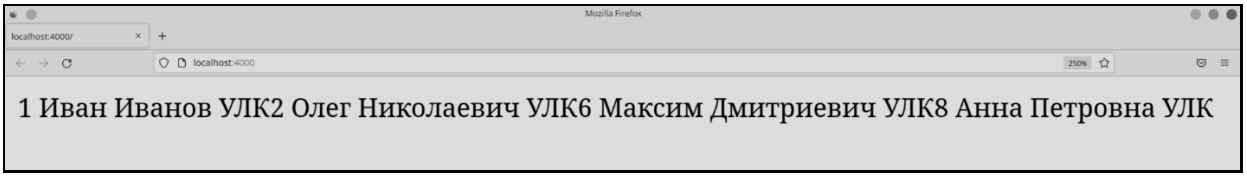


Рисунок 17 – Запрос «УЛК» для тестовой базы данных

После сравнения с исходной таблицей стало понятно, что программа работает правильно, ищет и выдает лишь нужные записи, согласно заданному условию в запросе. Полностью описанный программный код программы располагается ниже (рисунок 18).

Теперь, когда информация из таблицы «TSU» тестовой базы данных, передаются правильно, путем обработки записей программным кодом, можно приступить к решению о необходимости хранения данных. Осуществления сохранения данных, можно добиться путем создания еще одной таблицы, которая будет хранить обработанные записи и лишь те их поля, которые будут указаны разработчиком, что приведет к сохранности отчета, сформированного пользователем, но займет больше вычислительных ресурсов и памяти для хранения.

index.php

```php
<?php
try {
$dsn = 'firebird:dbname=/home/foxcat/FireBird/TSU.fdb;charset=UTF-8;';
$username = 'SYSDBA';
$password = '123';
$dbh= new PDO($dsn, $username, $password, [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]);
$SEARCH = 'УЛК';
$sql = 'SELECT * FROM TSU WHERE CORP='."'".$SEARCH."'";
$query = $dbh->query($sql);
while ($row = $query->fetch(\PDO::FETCH_OBJ)) {
echo $row-> ID, '  ', $row->NAME, ' ', $row->CORP;
}
$query->closeCursor();
}
catch (\PDOException $e) {
echo $e->getMessage();
}
?>
```

Рисунок 18 - Код phpPDO.php

Еще один вариант, это мгновенный и разовый вывод записей на экран, что не сохранит сформированный отчет, но сэкономит свободное место на сервере и оптимизирует по времени, работу алгоритма поиска записей. Из анализа преимуществ и недостатков методов, описанных выше, можно принять решение об использовании разового выведения, без хранения в промежуточной базе, так как из-за большого количества записей, следует экономить свободной пространство в памяти сервера и как можно быстрее обрабатывать полученную информацию.

Воспользуемся преимуществом языка PHP и поместим готовый код в наш проект. Для этого в одну из функций файла MainController.php поместим блок, который будет давать нам возможность исполнения нашей программы при открытии определенной веб-страницы, а также отображать результат программы в веб-приложении. Так как программа выводила на экран лишь объекты класса PDO без какого-либо разделения или упорядочивания, то мы будем использовать интерпретации языка разметки HTML внутри кода, тем самым самостоятельно задавая параметры отображения записей.

Для того, чтобы сохранить ввод данных с клавиатуры при поиске, мы воспользуемся формой для записи на странице поиска и методом post, который передавал данные нужные при отборе записей, соответствующих условию. Получаемые из текстового поля данные мы будем передавать с помощью метода request и помещать в переменную, которую подставим на место SEARCH, отвечающей за формирование условия нашего запроса, хранящегося в виде строки внутри SQL. Теперь запрос формируется согласно поставленной задаче, получаемые из базы данных объекты отображаются не как единая строка, а в виде записей с индивидуальными строками.

Контроллер после изменения (рисунок 19).

```php
    public function search_result(Request $request){
        $valid = $request->validate(['corp'=> 'required|min:1|max:100']);
        echo '<head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Результат поиска</title>
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">
        <style type="text/css">
                A { text-decoration: none;}
                A: hover {text-decoration: underline;
                color: dark;}
        </style>
    </head>
    <body class="bg-white">
        <div class="d-flex flex-column flex-md-row align-items-center p-3 px-md-4 mb-3 bg-white border-bottom shadow-sm">
        <h3 class="my-0 mr-md-auto font-weight-normal">TSU</h3>
        <nav class="my-2 my-md-0 mr-md-3">
            <a class="p-2 text-dark" href="/">Главная</a>
            <a class="p-2 text-dark" href="/search">Поиск</a>
        </nav>
        </div>';
        try {
            $dsn = 'firebird:dbname=/home/foxcat/FireBird/TSU.fdb;charset=UTF-8;';
            $username = 'SYSDBA';
            $password = '123';
            $dbh= new \PDO($dsn, $username, $password, [\PDO::ATTR_ERRMODE => \PDO::ERRMODE_EXCEPTION]);
            $sql = 'SELECT * FROM TSU WHERE CORP='."'".$request['corp']."'";
            $query = $dbh->query($sql);
            echo '<ul>';
            $i=0;
            while (($row = $query->fetch(\PDO::FETCH_OBJ))&&($i<1000)) {
            echo '<li>';
            echo '<pre>',$row->ID, "    ",$row->NAME, "    ",  $row->CORP,'</pre>';
            echo '</li>';
            $i++;
            }
            echo '<?ul>';
            $query->closeCursor();
            }
            catch (\PDOException $e) {
            echo $e->getMessage();
            }
        echo '</body> </html>';
    }
}

```

Рисунок 19 - Код контроллера MainController.php после добавления phpPDO.php

Разработку модуля можно считать завершенной, так как были выполнены все задачи, поставленные для реализации поиска событий на разных участках периметра системы контроля и управления доступом.

## 2.4 Тестирование приложения и анализ результатов

Результатом тестирования приложения, должно стать точное соответствие поставленной задаче. Разработанному проекту необходимо правильно работать в качестве модуля СКУД для отображения событий с разных участков периметра. Опробуем весь реализованный функционал веб-приложения.

Для начала протестируем переходы на различные страницы через кнопки навигации. Результаты перехода (рисунок 20) и (рисунок 21).


Рисунок 20 - Результат нажатия на кнопку «Главная»

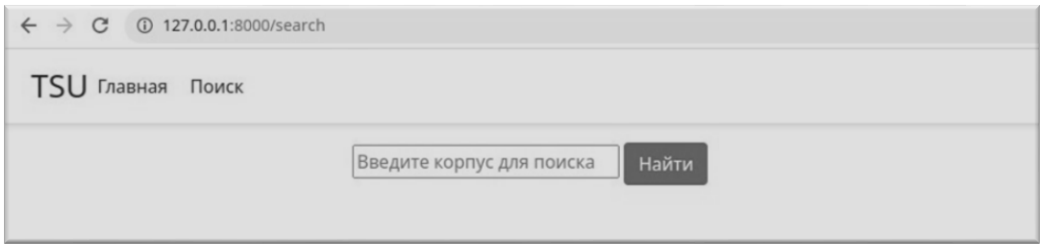


Рисунок 21 - Результат нажатия на кнопку «Поиск»

Навигация внутри проекта работает правильно. Перемещение по страницам работает, согласно заявленным данным.

Теперь на странице «Поиск» протестируем проверку введенных данных в поле для поиска. Оно должно выдавать ошибку в том случае, когда поле не заполнено, заполнено менее чем тремя символами или более чем 100. Результат проверки (рисунок 22).


Рисунок 22 - Результат ввода неправильных данных в поле поиска

Для всех трех неверных запросов сработало исключение, выдающее текст ошибки над полем для поиска. Валидация данных сработала правильно. Это указывает на то, что тестирование всей визуальной части модуля прошло успешно и результаты работы соответствуют поставленной задаче.

Приступим к тестированию блока модуля, отвечающего за поиск записей, согласно условию, заданному пользователем и их отображение на экране. Для этого применим те же входные данные, что и для проверки работоспособности алгоритма phpPDO.php, чтобы сравнить полученные данные с уже существующими и проверенными результатами. Для этого введем в поле для поиска по корпусу «УЛК». Результатом тестирование должны быть лишь записи, соответствующие заданному условию. Результат поиска по корпусу «УЛК» (рисунок 23).

После сравнения с исходной таблицей, становится понятно, что алгоритм сработал верно и выдал только те записи, которые соответствуют условию, что указывает на работоспособность алгоритма.

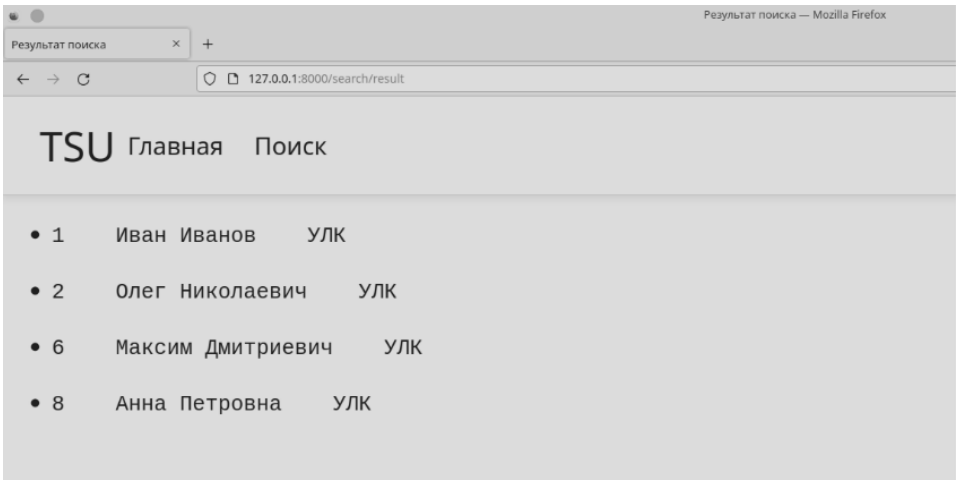


Рисунок 23 - Результат поиска по корпусу «УЛК»

Анализ результатов при тестировании модуля показал, что все данные полученные в ходе работы приложения, являются точными, интерфейс взаимодействия с пользователем отображается верно и выполняет требуемые

действия и алгоритмы, при использовании, а это означает, что разработанное приложение работает правильно и готово к использованию при поиске событий на разных участках периметра СКУД.

Вывод по второму разделу:

Во втором разделу производится программная реализация модуля системы контроля и управления доступом.

Создавались, рассматривались и тестировались методы реализации веб-страниц, контроллера, тестовой базы данных, записи которой имитируют события СКУД, сделана программа, проводящая поиск по базе данных. Все блоки были собраны в один программный продукт, который будет выполнять роль модуля системы контроля и управления доступом.

## Заключение

В данной пояснительной записке к выпускной квалификационной работе, были рассмотрены основные аспекты разработки модуля СКУД, который должен выводить события с разных участков периметра. Реализация производилось на тестовой упрощенной базе данных, записи которой имитируют события системы контроля и управления доступом.

В ходе исследования предметной области были указаны материалы, позволяющие ознакомиться с теоретическими данными и реализацией поиска информации по запросу с точки зрения математического моделирования, а также приложены иные материалы, способствующие пониманию того, как работает программная часть модуля, такие как блок-схема и граф, отображающий возможности взаимодействия программных компонентов продукта.

При описании программной реализации были рассмотрены и применены основные алгоритмы создания веб-приложений, баз данных на основе СУБД Firebird и реализация программы на скриптовом языке PHP, позволяющей манипулировать с базами данных, и их объектами. Так же в ходе разработки рассматривались дополнительные варианты реализации решения задач, с указанием преимуществ и недостатков, что позволяет адаптировать материалы данной работы для разработки вариативного программного продукта, отвечающего иным схожим требованиям по обработке информации базы данных.

Во время описания тестирования приложения и анализа его результатов, приводились примеры входных данных и их взаимодействия, чтобы показать правильность работы всех алгоритмов, блоков и подпрограмм. В ходе тестирования не было встречено ошибок, а результаты, полученные в ходе взаимодействия с программой в ходе анализа, оказались верными. Из всей информации полученной ранее следует, что программный

продукт работает правильно и готов к использованию в качестве модуля СКУД.

Подытожив все указанные данные, следует отметить, что для разработки данного программного продукта и модулей, для решения схожих задач, требуется знание и понимание разделов дискретной математики, математического анализа, аналитической геометрии, теории вероятностей и математической статистики для понимания и составления алгоритмов согласно математическим моделям поиска канонических видов, а также для реализации программной части нужно знать основные алгоритмы сортировки и поиска данных, понятия объективно-ориентированного программирования, язык разметки HTML и скриптовый язык PHP. Что позволяет применить и укрепить знания и навыки по данным дисциплинам, полученные в ходе обучения, а также развить их и получить новые, благодаря изучению и реализации практической части программного продукта.

## Список используемой литературы


1. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Паттерны объективно-ориентированного программирования. 1994. 395с.

2. ГОСТ №19.701-90 (ИСО 5807-85). Единая система программной документации схемы алгоритмов, программ, данных и систем. Москва Стандартформ. 2010. 24с.

3. Дронов В.А. Laravel. Быстрая разработка современных динамических Web-сайтов на PHP, MySQL, HTML и CSS. М.: БХВ-Петербург, 2018. - 507 с.

4. Кириченко А.В., Хрусталев А.А. HTML5 + CSS3. Основы современного WEB-дизайна. М.: Наука и техника, 2018. – 352 с.

5. Колисниченко Д. PHP и MySQL. Разработка Web-приложений. – СПб.: БХВ-Петербург, 2017. – 640 с.

6. Краткое руководство по миграции на Firebird 4.0. URL: https://www.ibase.ru/files/firebird/fb4migrationshort.pdf

7. Кроудер Д. Создание веб-сайта для чайников. 2019. 336 с.

8. Кузнецов М.А., Нгуен Т.Т. Математические модели информационного поиска WEB-Ресурсов./Прикаспийский журнал: управление и высокие технологии №2 (22). 2013. URL: https://hi-tech.asu.edu.ru/files/2(22)/25-30.pdf

9. Кузнецов М.В., Симдянов И.В. Объективно-ориентированное программирование на PHP. 2007. 608с.

10. Кучукова Н.Н., Вершков Н.А. Математическая модель подсистемы поиска и ранжирования документов в информационно-поисковых сетях //Журнал «Экономика. Информатика» 2018. том 45.

11. Магда Ю.С., Современные микроконтроллеры. Архитектура, программирование, разработка устройств. – М.: ДМК Пресс, 2017. – 224 с.

12. Маклафлин, Б. PHP и MySQL. Исчерпывающее руководство / Б. Маклафлин. - М.: Питер, 2017. - 379 с.

13. Матросов, А.В. HTML 4.0 / А.В. Матросов. - М.: БХВ-Петербург, 2022. - 735 с.

14. Настройка и использование PDO – расширения PHP Data Objects для работы с базами данных. 2017. URL: https://tproger.ru/translations/how-to-configure-and-use-pdo/

15. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. 2018. 816 с.

16. Основы работы с расширением PDO. URL: http://codeharmony.ru/materials/137

17. Симонов Д., Еманов Д., Кузьменко Д., Ковязин А. Руководство разработчика Firebird. 2018. 279 с. URL: https://www.ibase.ru/files/firebird/fbdevguide30.pdf

18. Тузовский А.Ф. Проектирование и разработка web-приложений. Учебное пособие. – М.: Юрайт, 2017. – 218 с.

19. Филиппов Д., Карпейкин А., Ковязин А., Кузьменко Д., Симонов Д., Vinkenoog P., Еманов Д., Rotteveel M. Руководство по языку SQL СУБД Firebird 4.0. 2021. 903 с.

20. Формула полной вероятности. Теорема гипотез (формула Байеса). URL: https://studbooks.net/2186302/matematika_himiya_fizika/vvedenie

21. Carlos H. Cantu Migration Guide to Firebird 4.0. 2021. -206 с.

22. Fundamentals of PHP PDO. URL: https://www.phptutorial.net/php-pdo/

23. Hardik, Dangar Learning Laravel 4 Application Development / Hardik Dangar. - М.: 2021. - 256 с.

24. Kevin Tatroe, Peter MacIntyre. Programming PHP: Creating Dynamic Web Pages. 2020. -544 с.

25. Meher Krishna Patel. HTML, CSS, Bootstrap, Javascript and jQuery 2017. URL: http://englishonlineclub.com/pdf/HTML,%20CSS,%20Bootstrap,%20Javascript%20and%20jQuery%20[EnglishOnlineClub.com].pdf