

Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерного проектирования
Кафедра проектирования информационно-компьютерных систем
Дисциплина «Структуры и базы данных»

«К ЗАЩИТЕ ДОПУСТИТЬ»
Руководитель курсового проекта
магистр техники и технологии
_____._____.2020 А.В. Шелест

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему:
«БАЗА ДАННЫХ ДЛЯ ПОДДЕРЖКИ РАБОТЫ КАФЕДРЫ»

БГУИР КР1-39 03 02 036 ПЗ

Выполнил студент группы 813801
ШАМРЕЙ Андрей Станиславович

(подпись студента)

Курсовой проект представлен на
проверку _____._____.2020

(подпись студента)

Минск 2020

РЕФЕРАТ

БГУИР КП 1-39 03 02 036 ПЗ

Шамрей, А.С. База данных для поддержки работы кафедры : пояснительная записка к курсовому проекту / А.С. Шамрей. – Минск : БГУИР, 2020. – 40 с.

Пояснительная записка 40 с., 17 рис., 10 источников, 3 приложения

Цель проектирования: проектирование базы данных, используемой для поддержки работы кафедры.

Методология проведения работы: в процессе решения поставленных задач использованы принципы системного подхода, теория проектирования реляционных баз данных, методы проектирования баз данных с использованием CASE-средств.

Результаты работы: выполнен анализ предметной области и информационных потребностей пользователей, спроектирована реляционная база данных для работы кафедры, разработано веб-приложение для работы с базой данных.

Курсовой проект состоит из расчётно-пояснительной записки, графической и программной части. В расчётно-пояснительной записке приводится общая характеристика базы данных, раскрываются задачи и требования базы данных.

Приложение обеспечивает управление данными, необходимыми для работы кафедры: личной информацией и расписанием преподавателей. Также приложение обеспечивает корректный просмотр необходимых данных сотрудниками кафедры: расписание преподавателей, объявления о консультациях, документы кафедры.

Область применения результатов: приложение может быть использовано для обеспечения комфортного обмена информацией среди сотрудников кафедры университета, тем самым создавая приятные условия работы.

СОДЕРЖАНИЕ

Введение.....	6
1 Анализ предметной области и ее формализация для проектирования базы данных	8
1.1 Описание предметной области	9
1.2 Анализ информационных потребностей пользователей и предварительное описание запросов	9
1.3 Определение требований и ограничений к базе данных с точки зрения предметной области	10
1.4 Постановка решаемой задачи	11
2 Проектирование базы данных для основного вида деятельности рассматриваемой предметной области	13
2.1 Разработка инфологической модели предметной области базы данных	13
2.2 Выбор и обоснование используемых типов, данных и ограничений (доменов).....	15
2.3 Проектирование запросов к базе данных	16
2.4 Программная реализация и документирование базы данных	18
2.5 Проектирование разрабатываемого приложения	19
3 Применение разработанной базы данных	21
3.1 Руководство пользователя.....	21
3.2 Администрирование базы данных	26
3.3 Реализация клиентских запросов.....	28
3.4 Обоснование и реализация механизма обеспечения безопасности и сохранности данных	29
Заключение	31
Список использованных источников	32
Приложение А (обязательное) Отчёт о проверке на уникальность в системе «Антиплагиат»	33
Приложение Б (обязательное) Скрипт генерации базы данных	34
Приложение В (Листинги программного кода)	37
Ведомость курсового проекта.....	40

ВВЕДЕНИЕ

Система управления базами данных (СУБД) – комплекс программных и лингвистических средств общего или специального назначения. СУБД позволяет создавать базу данных и манипулировать данными (вставлять, обновлять, удалять и выбирать). Система обеспечивает безопасность, надёжность хранения и целостность данных, а также предоставляет средства для администрирования базы данных.

База данных – совокупность структурированных взаимосвязанных данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования, независимая от прикладных программ. База данных является информационной моделью предметной области. Цель базы данных – помочь людям и организациям вести учет, осуществлять информационную поддержку при принятии решений. Обращение к базам данных осуществляется с помощью системы управления базами данных [1].

В последние годы бурное развитие новых технологий сделали актуальным использование баз данных в организациях различных форм собственности и направлений деятельности. Успехи в исследованиях и разработке баз данных стали основой фундаментальных разработок коммуникационных систем, транспорта и логистики, финансового менеджмента, методов доступа к научной литературе, а также многих гражданских и военных приложений.

Базы данных лежат в основе информационных систем, и это коренным образом изменило характер работы многих предприятий и организаций. В контексте данного курсового проекта рассматривается база данных для поддержки работы кафедры.

Актуальность работы обусловлена тем, что для поддержания высокого уровня эффективности работы кафедры необходимы инструменты, соответствующие времени. В настоящее время бурно развиваются различные интернет-площадки для общения и передачи информации, и, соответственно, всё больше людей пользуются данными ресурсами. Для обеспечения более комфортных условий передачи информации среди сотрудников кафедры данная работа предлагает использование веб-приложения, позволяющего просматривать объявления, расписание и контактную информацию всех сотрудников кафедры. При этом разработанное приложение должно иметь приятный и интуитивно понятный интерфейс, что позволит без особого труда разобраться в его работе любому пользователю.

Целью данного курсового проекта является изучение теории и практическое использование ее в создании эффективной и безопасной базы данных для поддержания работы кафедры, работе с СУБД, применении языка запросов *SQL*, написании пользовательского приложения для манипуляции с базой данных, что, несомненно, является актуальной задачей на сегодняшний день.

Основные задачи, необходимые для достижения поставленной цели:

- анализ предметной области и ее информационных потребностей;
- изучение основных языков программирования и разметки, таких как *HTML, CSS, JavaScript и PHP*;
- изучение языков запросов баз данных на примере *SQL*;
- приобретение навыков организации взаимодействия разработанных баз данных и ПО;
- изучение основных задач администрирования и анализа баз данных для формирования эффективных и безопасных систем хранения информации.

В результате должны получить схему реляционной базы данных и веб-приложение, реализующее механизмы работы с данными в базе данных.

Для проверки уникальности работы используется интернет-ресурс antiplagiat.ru. Результат проверки работы на уникальность приведен в приложении А.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ЕЁ ФОРМАЛИЗАЦИЯ ДЛЯ ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ

Предметная область – это целенаправленная первичная трансформация картины внешнего мира в некоторую умозрительную картину, определенная часть которой фиксируется в информационной системе в качестве алгоритмической модели фрагмента действительности.

Анализ предметной области предполагает составление описания предметной области, которое подразумевает формулирование и анализ требований, предъявляемых к содержанию и процессу обработки данных всеми известными и потенциальными пользователями базы данных. На этапе системного анализа необходимо провести подробное словесное описание информационных объектов предметной области и реальных связей, которые присутствуют между описываемыми объектами. Системный анализ является наиболее трудным и длительным этапом процесса проектирования.

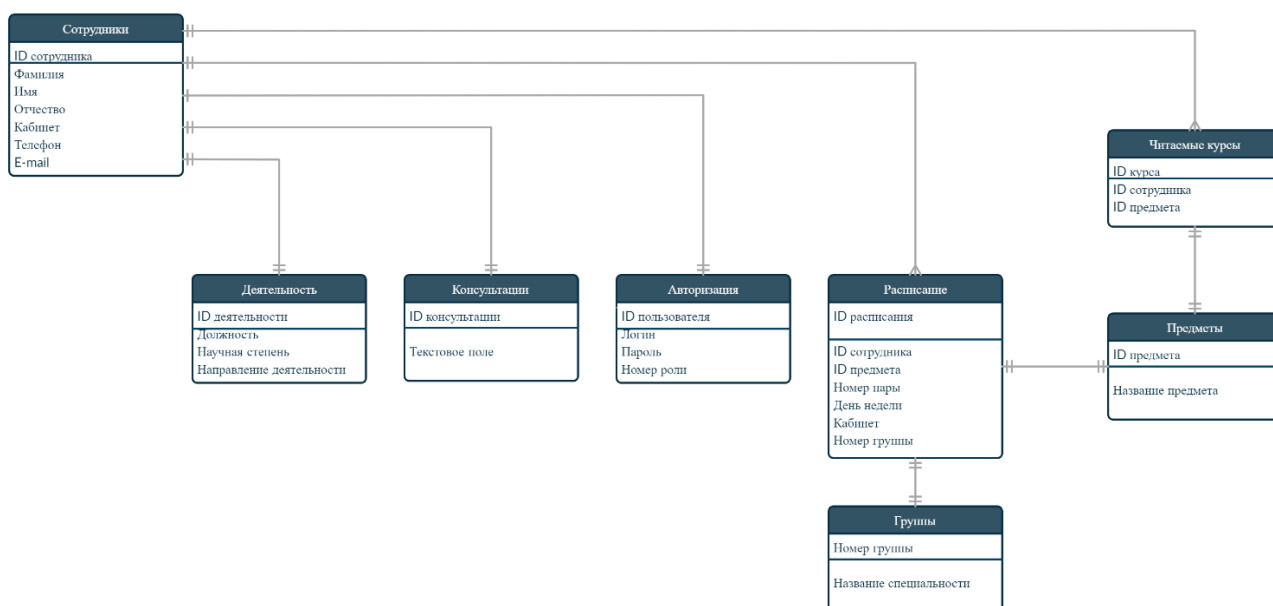


Рисунок 1.1 – ER-модель предметной области

В результате системного анализа должны быть сформулированы: подробное описание информации об объектах предметной области и информационных процессах; конкретные задачи, которые будут решаться базой данных с кратким описанием алгоритма решения; описание документов, которые должны генерироваться в системе; описание документов, которые служат основанием для заполнения базы данных.

1.1 Описание предметной области

Задача проекта – создание информационной поддержки учебного процесса и организационной деятельности на кафедре университета. Различают две группы сотрудников:

- профессорско-преподавательский состав;
- учебно-вспомогательных состав.

База данных должна содержать штатное расписание преподавателей кафедры с указанием места и времени проведения занятия, а также включать архив сведений о сотрудниках.

База данных должна предоставлять сотруднику кафедры возможность добавления информации о проводимых дополнительных занятиях, а также о возможных объявлениях и изменениях в расписании.

Эта база данных должна содержать сведения о сотрудниках университета: личных данных сотрудников (фамилия, имя, отчество, дата рождения, мобильный телефон, электронная почта и так далее), их трудовой деятельности (должности, научные степени, области научных исследований, читаемые курсы и так далее). Кроме того база данных кафедры должна хранить информацию о группах студентов и специальностях, на которых преподают сотрудники кафедры. *ER*-модель предметной области показан на рисунке 1.1.

1.2 Анализ информационных потребностей пользователей и предварительное описание запросов

Рассмотрим университет, подготавливающий специалистов разного профиля. На кафедре университета различаются две группы сотрудников: профессорско-преподавательский состав и учебно-вспомогательных состав.

Необходимость разработки базы данных обусловлена стремлением структурировать информацию о сотрудниках и облегчить получения сотрудниками информации о расписании проводимых занятий.

Сайт кафедры предоставит возможность структурировать расписание преподавателей, а также даст возможность беспрепятственно получать информацию о расписании занятий преподавателей. Также будет предоставлен доступ к хранилищу документов и файлов кафедры.

Данной базой данных будут пользоваться сотрудники кафедры университета и администратор. Сотрудники кафедры будут иметь возможность просматривать личные данные и расписание преподавателей кафедры, редактировать свои личные данные, просматривать документы кафедры, а также размещать дополнительную информацию на личной страничке. Администратор будет иметь полный доступ ко всем сущностям системы: редактирование данных о сотрудниках, удаление сотрудников из базы данных, добавление сотрудников в базу данных. кроме того, администратору будут доступны все возможности обычного пользователя.

Для визуального отображения информационных потребностей используется диаграмма вариантов использования, показанная на рисунке 1.2.

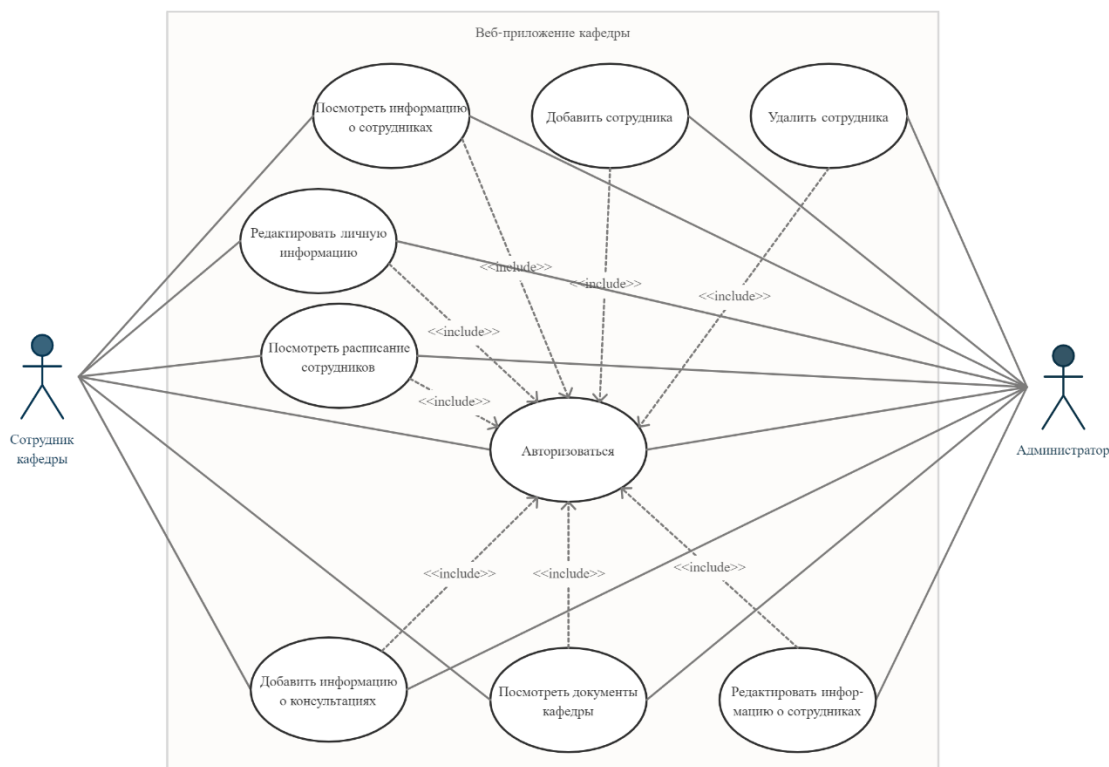


Рисунок 1.2 – Диаграмма вариантов использования веб-приложения кафедры

Данная диаграмма позволяет наглядно отобразить информационные потребности в проектируемой системе.

1.3 Определение требований и ограничений к базе данных с точки зрения предметной области

Информационная система, как и любой другой инструмент, должна иметь свои характеристики и требования, в соответствии с которыми можно было бы определить ее функциональность и эффективность. Ниже приведены основные требования и ограничения, предъявляемые к проектируемой базе данных.

Требование уникальности записей распространяется на следующие атрибуты: уникальный номер пользователя *user_id* для аутентификации; *role_id* для определения типа пользователя; *group_number* для указания номера группы; *speciality_name* для указания названия специальности. Атрибуты *user_id* и *role_id* должны включать только цифры.

Обязательность указания имеют следующие атрибуты: *first_name*, *last_name* и *father_name* для указания ФИО; *login* и *pass* для указания логина и пароля пользователя.

1.4 Постановка решаемой задачи

В данном курсовом проекте требуется изучить особенности и структуру реляционных баз данных. Реляционные базы данных основаны на теоретико-множественной реляционной даталогической модели. Все данные представляются в виде связанных между собой таблиц, разбитых на строки и столбцы. Каждая таблица должна иметь первичный ключ – поле или набор полей, содержимое которых однозначно определяет запись в таблице и отличает ее от других. Связь между двумя таблицами обычно образуется при добавлении в первую таблицу поля, содержащего значение первичного ключа второй таблицы.

Для создания базы данных необходимо проанализировать имеющуюся информацию о уже существующих, создать на ее основе необходимые таблицы, установить определенные связи между ними. Поэтому задачи, которые нужно будет решить в ходе курсового проекта, заключаются в разработке связей между таблицами базы данных, устранении избыточности в таблицах, их обработке и редактировании [2].

Эффективность работы любой организации во многом зависит от комфорта и условий работы сотрудников. Работники должны иметь быстрый доступ к расписанию сотрудников кафедры, а также к документам и публикациям кафедры.

Работа кафедры связана с накоплением большого количества информации о личных данных сотрудников. Традиционно информация хранится на бумажных носителях. При этом трудно осуществить быстрый отбор нужных данных.

Создание веб-приложения с взаимодействием с базой данных позволит сократить время на обработку информации, произойдет сокращение затрат на обработку информации, уменьшатся затраты времени на поиск необходимой информации, улучшится качество контроля и учета обрабатываемой информации, повысится эффективность работы всего состава кафедры.

Необходимость проектирования базы данных обусловлена стремлением структурировать информацию о сотрудниках и облегчить получения сотрудниками информации о расписании проводимых занятий

Целью данного проекта является проектирование базы данных для удобства хранения и использования информационной базы о сотрудниках кафедры, автоматизации получения и анализа данных о занятости сотрудников кафедры. Необходимо не только упорядочить информацию, но и упростить процессы ее анализа и принятия необходимых управленческих решений.

Целью всех выполняемых действий являются:

- автоматизировать процесс получения информации о сотрудниках;
- сократить время поиска необходимых данных;
- оптимизировать хранение информации, необходимой для деятельности кафедры.

Задачами данного проекта являются:

- выявить необходимость автоматизации информационных потоков процессов деятельности кафедры;
- проектирование базы данных *MySQL*;
- создание веб-приложения для проверки работоспособности спроектированной базы данных.

Главная задача моделируемой системы – сохранение в базе данных всех необходимых сведений о сотрудниках кафедры и их расписании; представление данных в удобном для пользователя виде.

Требования к целостности данных: данные в базе данных в любой момент времени должны быть правильными и непротиворечивыми.

Требования к безопасности: доступ к системе должен быть осуществлён после аутентификации пользователя, на основе прав и ролей пользователей.

Кроме того, клиентское приложение должно иметь дружелюбный и приятный интерфейс.

2 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ ДЛЯ ОСНОВНОГО ВИДА ДЕЯТЕЛЬНОСТИ РАССМАТРИВАЕМОЙ ПРЕДМЕТНОЙ ОБЛАСТИ

Проектирование базы данных является длительным и трудоёмким процессом. Это одна из наиболее сложных и ответственных задач, связанных с созданием информационной системы. Разработка базы данных и прикладного программного обеспечения для поддержания работы кафедры университета состоит из проектирования базы данных и разработки прикладного программного обеспечения.

Процесс проектирования включает в себя следующие этапы: концептуальное проектирование, логическое проектирование, физическое проектирование [3].

В этом разделе будет рассмотрен способ создания полноценной базы данных.

2.1 Разработка инфологической модели предметной области базы данных

Инфологическая модель данных – обобщенное неформальное описание создаваемой базы данных, выполненное с использованием естественного языка, математических формул, таблиц, графиков и других средств, понятных всем людям, работающим над проектированием базы данных. Она является человеко-ориентированной моделью, которая полностью независима от физических параметров среды хранения данных [4].

Цель инфологического моделирования – обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных. Храниться данные будут в диаграмме «сущность – связь», которая будет создана с помощью системы управления базами данных.

Модель «сущность – связь» является графическим средством представления объектов рассматриваемой предметной области, их характеристик и отношений между объектами.

Основными элементами инфологической модели являются сущности, связи между ними и их атрибуты.

Сущность – любой конкретный или абстрактный объект в рассматриваемой предметной области. Сущности – это базовые типы информации, которые хранятся в базе данных.

Атрибут – это свойство сущности в предметной области. Его наименование должно быть уникальным для конкретного типа сущности.

Связь – это некоторая ассоциация между двумя сущностями. Связи представляют собой соединения между частями базы данных [5].

Диаграмма «сущность – связь» (*ER*-диаграмма) позволяет графически представить все элементы информационной модели согласно простым, интуитивно понятным, но строго определенным правилам – нотациям.

Прежде чем проектировать базу данных, необходимо привести данные к третьей нормальной форме и ответить на два вопроса:

- какой атрибут будет являться первичным ключом;
- какие атрибуты будут являться внешними ключами.

На этапе концептуального проектирования, имея словесное описание предметной области, предусматривается выполнение таких работ как:

- идентификация объектов предметной области, их атрибутов и первичных ключей;
- идентификация отношений между объектами и указание мощности этих связей;
- построение концептуальной схемы базы данных на основе модели «сущность – связь».

В данной работе основными информационными объектами предметной области являются: сотрудники, деятельность, консультации, авторизация, расписание, читаемые курсы, роли, группы, предметы.

Между ними можно установить следующие логические связи:

- связь сотрудники-деятельность мощностью «один-к-одному». Один сотрудник может заниматься одной деятельностью.
- связь сотрудники-консультации мощностью «один-к-одному». Один сотрудник может иметь только одно поле для ввода дополнительной информации на сайте.
- связь сотрудники-авторизация мощностью «один-к-одному». Один сотрудник может иметь только один аккаунт в веб-приложении кафедры.
- связь сотрудники-расписание мощностью «один-ко-многим». Один сотрудник может вести несколько пар в день и у разных групп.
- связь расписание-группы мощностью «один-к-одному». Одному номеру группы может соответствовать только одно название специальности.
- связь расписание-предметы мощностью «один-к-одному». Одному номеру предмета может соответствовать только одно название предмета.
- связь сотрудники-читаемые курсы мощностью «один-ко-многим». Один сотрудник может читать несколько курсов.
- связь читаемые курсы-предметы мощностью «один-к-одному». Один курс соответствует одному названию предмета.

На этапе логического проектирования разрабатывается логическая схема базы данных, обосновывается выбранная методика проектирования базы данных, описывается процесс проектирования логической схемы базы данных и этапы ее нормализации.

Также на этапе логического проектирования концептуальная модель отображается в логическую с учетом выбранной реляционной модели данных по правилам преобразования.

Каждый объект концептуальной модели отображается в таблицу базы данных. Результат логического проектирования изображается на *ER*-диаграмме, показанной на рисунке 2.1.

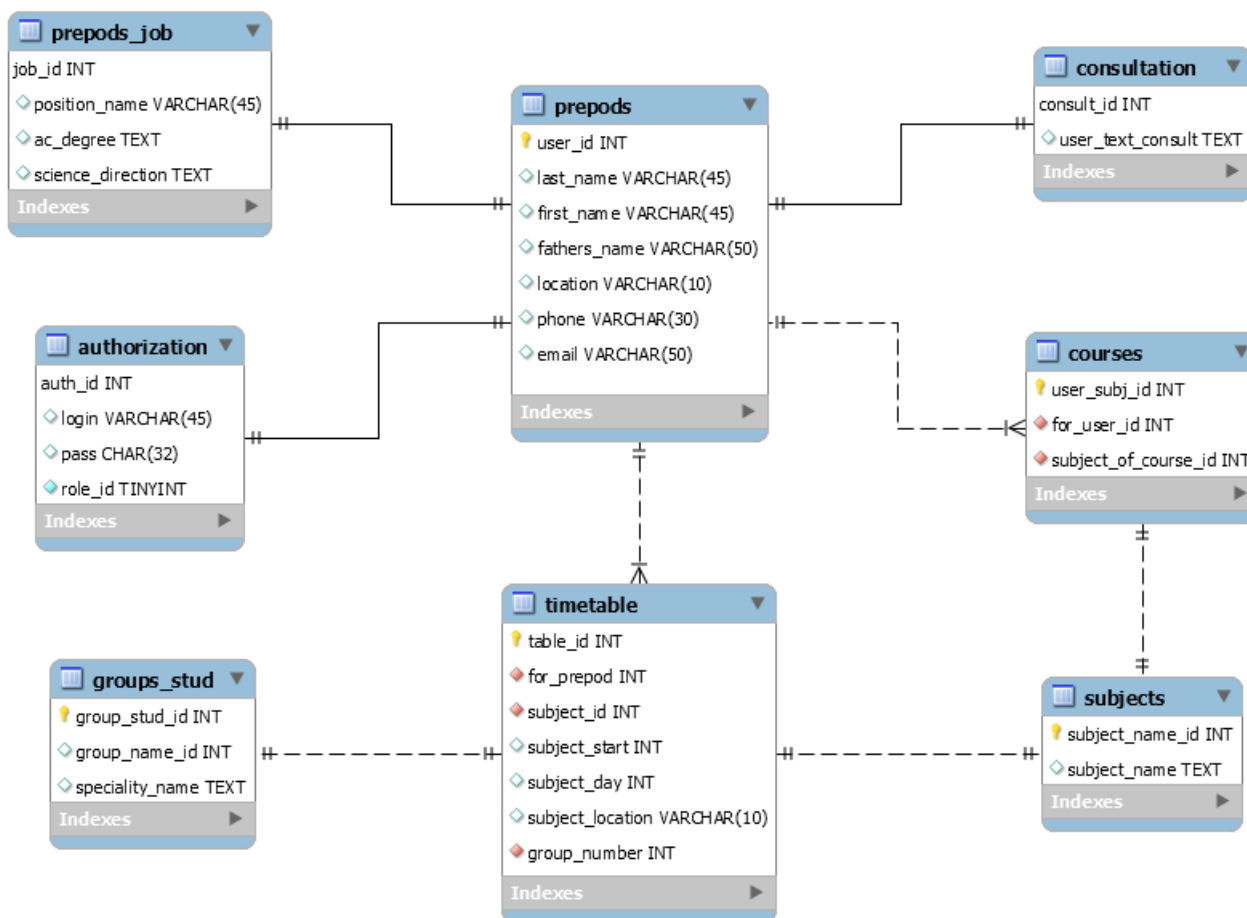


Рисунок 2.1 – Логическая схема проектируемой базы данных

Для достижения минимальной избыточности данных в базе данных необходимо произвести нормализацию таблиц (отношений) базы данных. На практике достаточно, чтобы отношение находилось в 3 нормальной форме. Отношение удовлетворяет третьей нормальной форме, тогда и только тогда, когда отношение находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых [5].

Каждая таблица спроектированной базы данных проверена на соответствие требованиям 3 нормальной формы.

2.2 Выбор и обоснование используемых типов, данных и ограничений (доменов)

Физическое проектирование является третьим и последним этапом создания проекта базы данных, при выполнении которого принимаются решения о способах реализации разрабатываемой базы данных.

Приступая к физическому проектированию базы данных, прежде всего необходимо выбрать конкретную целевую СУБД, так как физическое проектирование неразрывно связано с конкретной СУБД.

В данной работе используется СУБД *MySQL*. *MySQL* – это популярная свободная реляционная система управления базами данных. *MySQL* базируется на языке *SQL* и поддерживает многочисленные возможности.

Важным этапом проектирования базы данных является выбор и обоснование типов данных. Любые данные, помещаемые в столбец, должны отвечать этому типу данных. Тип данных определяет, какие значения могут храниться в столбце и сколько они будут занимать места в памяти.

MySQL поддерживает несколько типов столбцов, которые можно разделить на три категории: числовые типы данных, типы данных для хранения даты и времени и символьные (строковые) типы данных [6].

При проектировании базы данных были использованы следующие типы данных.

Тип данных *CHAR*, который представляет строку фиксированной длины, при этом, если указано меньше символов, чем задано, то строка дополняется пробелами и в итоге все равно будет занимать заданное количество символов;

Тип данных *VARCHAR*, который представляет строку переменной длины. Хранимая строка будет занимать именно столько места, сколько необходимо, что позволит сократить память, выделенную для этой строки, если в ней указано символов меньше, чем задано. Данный тип выбран для таких атрибутов сущностей, размер которых не должен превышать порогового значения (фамилия, имя, отчество, должность, логин и так далее);

Тип данных *TEXT*, который представляет текст длиной до 65 килобайт и используется для записи в строку символьной информации, превосходящей длину в 255. Данный тип выбран для таких атрибутов сущностей, у которых нет количество символов не ограничено (текстовое поле, направление научной деятельности и так далее);

Тип данных *INT*, который представляет целые числа от -2147483648 до 2147483647, занимает 4 байта и используется для нумерации элементов и записи числовых данных. Данный тип выбран для таких атрибутов сущностей, которые представляют собой нумерацию записей в сущности (*id* сотрудник, *id* расписания, *id* деятельности и так далее), либо численные значения (номер группы, номер пары, день недели и так далее);

Тип данных *TINYINT*, который представляет целые числа от -127 до 128, занимает 1 байт и используется для нумерации небольшого количества элементов (например, *id* роли – 1 либо 2).

2.3 Проектирование запросов к базе данных

Запрос строится на основе одной или нескольких взаимосвязанных таблиц, позволяя комбинировать содержащуюся в них информацию. Запрос

позволяет выбрать необходимые данные из одной или нескольких взаимосвязанных таблиц и получить результат в виде новой таблицы. Полученная таблица может использоваться в качестве источника данных в следующих запросах, формах, отчетах, страницах доступа к данным. Через запрос можно производить вычисление, изменение данных в таблицах, добавление и удаление записей и многое другое.

Для работы с проектируемой базой данных в большинстве случаев достаточно паттерна *CRUD*. *CRUD* – акроним, обозначающий четыре базовые функции, используемые при работе с базами данных: создание (*create*), чтение (*read*), модификация (*update*) и удаление (*delete*).

В *SQL* этим функциям операциям соответствуют операторы *Insert* (создание записей), *Select* (чтение записей), *Update* (редактирование записей), *Delete* (удаление записей) [7].

Оператор *INSERT INTO* предназначен для добавления данных в таблицу. Синтаксис оператора: *INSERT INTO table_name ([column_name, ...]) VALUES (expressions, ...)*.

Оператор *UPDATE* предназначен для обновления данных таблицы. Эта команда имеет следующий синтаксис: *UPDATE table_name SET expression [WHERE condition]*.

Оператор *DELETE* предназначен для удаления данных из таблицы. Синтаксис оператора: *DELETE from table_name [WHERE condition]*. Если не будут указаны дополнительные условия с помощью команды *WHERE*, то из таблицы будут удалены все значения.

Для вывода данных из таблицы используется оператор *SELECT*. Синтаксис оператора: *SELECT column_list FROM table_name [WHERE condition GROUP BY expression HAVING condition ORDER BY expression]*.

Рассмотрим команды вывода данных на примере проектируемой базы данных. Чтобы вывести список сотрудников и их номера телефонов используем следующий запрос:

```
SELECT first_name, last_name, phone FROM prepods;
```

Также можно вывести список сотрудников, *id* которых больше 1. Необходимый для этого запрос имеет вид:

```
SELECT * FROM prepods WHERE user_id > 1;
```

Чтобы вывести список сотрудников, отсортированный по имени, используется следующий запрос:

```
SELECT * FROM prepods ORDER BY first_name;
```

Чтобы вывести количество сотрудников, работающих в каждом кабинете, можно использовать следующий запрос:

```
SELECT location, count(first_name) AS num_of_prepos FROM prepos GROUP BY location;
```

Чтобы вывести только те кабинеты, в которых работает больше одного сотрудника, можно использовать следующий запрос:

```
SELECT location, count(first_name) AS num_of_prepos FROM prepos GROUP BY location HAVING count(first_name) > 1;
```

Запросы позволяют решать многие задачи, не прибегая к программированию. Например, представлять данные в агрегированном виде, производить вычисления над полями базы данных, группировать записи и находить для полей итоговые значения с помощью статистических функций: *Sum*, *Avg*, *Max*, *Min*, *Count* и другие.

2.4 Программная реализация и документирование базы данных

Для визуального проектирования баз данных была выбрана программа *MySQL Workbench*, позволяющая интегрировать визуальное проектирование, моделирование, создание и эксплуатацию базы данных в единое бесшовное окружение для системы баз данных *MySQL*.

Для удобства сперва была разработана ER-диаграмма спроектированной базы данных, затем были установлены связи и столбцы в таблицах. На основе ER-диаграммы и схемы в среде *MySQL Workbench*, используя программные средства интерпретации визуального проектирования в автоматически генерируемый *SQL*-скрипт была создана база данных заданной предметной области – база данных для поддержки работы кафедры.

Чтобы подключить существующую базу данных к веб-серверу, необходимо подключить *MySQL*-соединение, открывающее доступ к базе данных посредством сетевых технологий. Стандартно *MySQL* разворачивает сервер с базой данных на локальном IP-адресе 127.0.0.1 (или *localhost*) и располагает оконечные точки для обмена сообщениями на порте 3306 [8].

Для подключения к локальному серверу с базой данных необходимо ввести личные данные администратора в графическом пользовательском интерфейсе *MySQL Workbench* или выполнить команду *mysql -u root -p* в приветственной строке интерфейса командной строки.

Таким образом, спроектированная ранее визуально база данных была разработана программно и подключена к локальному серверу внутренними средствами утилиты *MySQL Workbench*, предоставляющей графический пользовательский интерфейс СУБД *MySQL*.

Программная реализация база данных на языке запросов представлена в Приложении Б.

2.5 Проектирование разрабатываемого приложения

В рамках курсового проекта, для проверки работоспособности спроектированной базы данных было предложено разработать веб-приложение, реализующее клиент-серверное взаимодействие.

Клиент – это оконечный пользователь приложения, который видит результаты его работы непосредственно в окне браузера на экране персонального компьютера, смартфона или ноутбука. Сервер – вычислительный блок, расположенный в сети Интернет или развёрнутый локально на компьютере администратора веб-приложения, обращающийся к базе данных и обменивающийся *HTTP*-запросами с клиентской стороной, то есть браузером.

Первичное изучение подходов веб-разработки и анализ аналогичных существующих веб-систем, показал, что с архитектурной точки зрения разрабатываемая система будет состоять из нескольких слабо зависимых частей, каждая из которых будет служить определенной цели, и соответственно, иметь свой набор технологий. К таким частям относят:

- внешний интерфейс (*Frontend*);
- серверное *API* (*Backend*).

В качестве серверной стороны веб-приложения для обеспечения работоспособности кафедры университета был выбран язык программирования *PHP*, реализующий принципы ООП.

Само веб-приложение будет разработано на языке гипертекстовой разметки *HTML*. Для создания приятного и эргономичного пользовательского интерфейса используется формальный язык описания внешнего вида документа *CSS*. Также будет использован язык программирования, с помощью которого страницы сайтов становятся интерактивными – *JavaScript*.

В процессе разработки были построены следующие UML-диаграммы: диаграмма вариантов использования, диаграмма классов, диаграмма состояний.

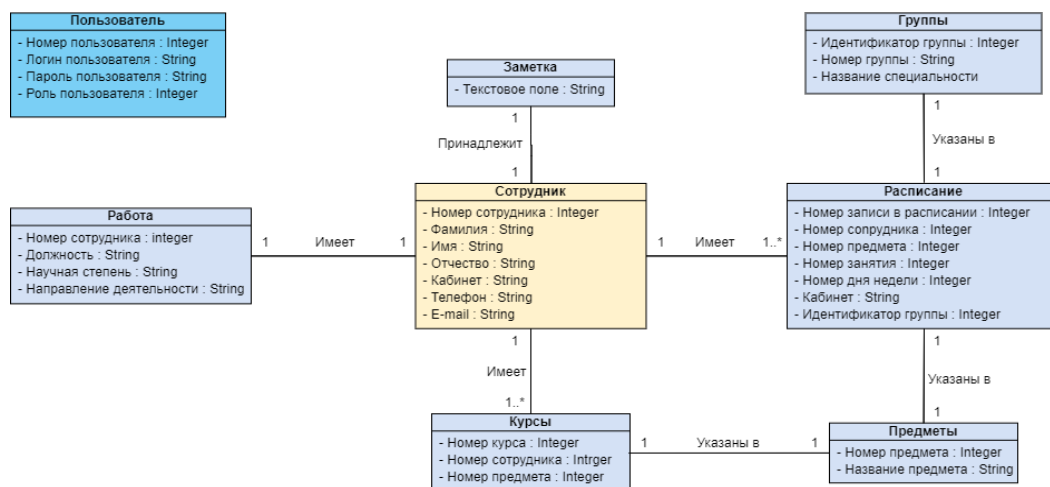


Рисунок 2.2 – Диаграмма классов

Диаграмма классов – структурная диаграмма, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними. В данном проекте диаграмма классов использована в качестве словаря предметной области. Данная диаграмма представлена на рисунке 2.2.

Диаграмма состояний – графическое изображение всех возможных состояний системы. Главное предназначение этой диаграммы – описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла. Диаграмма состояний показывает все возможные состояния, в которых может находиться объект, а также процесс смены состояний в результате внешнего влияния. Данная диаграмма представлена на рисунке 2.3.

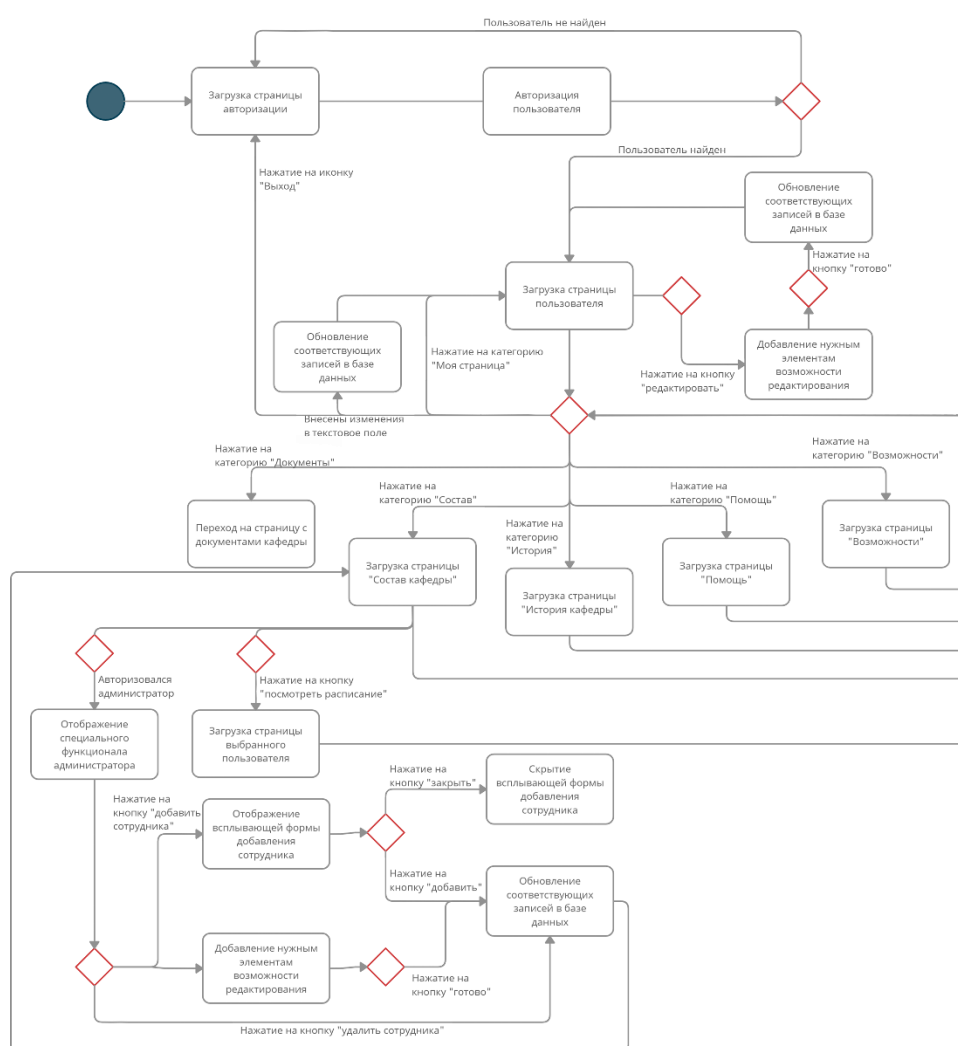


Рисунок 2.3 – Диаграмма состояний

Диаграмма состояний используется аналитиками для описания последовательности переходов объекта из одного состояния в другое.

3 ПРИМЕНЕНИЕ РАЗРАБОТАННОЙ БАЗЫ ДАННЫХ

3.1 Руководство пользователя

Для начала работы с системой пользователю необходимо авторизоваться: ввести логин и пароль от своей учетной записи. Пример авторизации показан на рисунке 3.1.

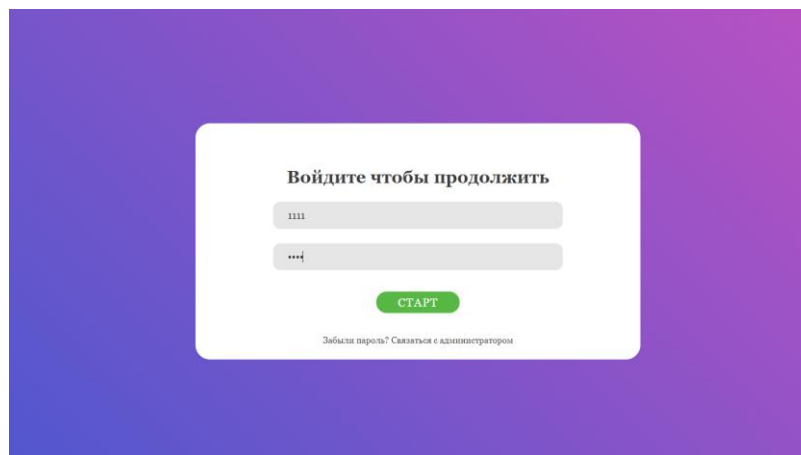


Рисунок 3.1 – Форма авторизации

После авторизации пользователь получает доступ к разрешенным ему разделам и возможностям. На рисунке 3.2 показан интерфейс приложения для сотрудника кафедры.

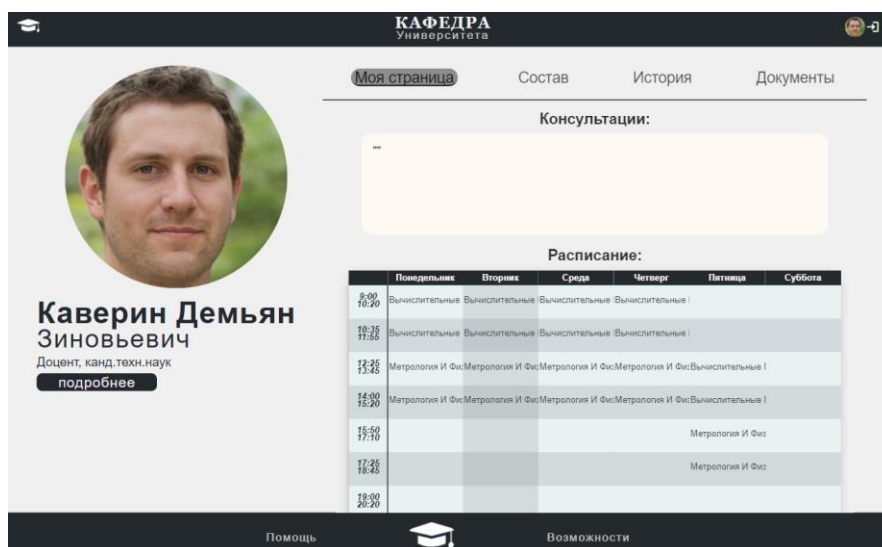





Рисунок 3.2 – Интерфейс приложения. Главная страница

При нажатии на кнопку «подробнее», раскрывается информация о сотруднике. Результат показан на рисунке 3.3.


КАФЕДРА
 Университета





**Каверин Демьян
Зиновьевич**
 Доцент, канд. техн. наук
[скрыть](#)
 Аудитория: 37-1
 Телефон: +375297682610
 E-mail: o@outlook.com

Читаемые курсы:

- Метрология и физические основы измерений
- Вычислительные методы и компьютерная алгебра

Направление научной деятельности:
 Синтез и исследование свойств
 монокристаллов сложных полупроводниковых
 соединений.

[редактировать](#)


Моя страница
Состав
 История
 Документы


Консультации:

Расписание:

	Понедельник	Вторник	Среда	Четверг	Пятница	Суббота
9:00 10:20	Вычислительные	Вычислительные	Вычислительные	Вычислительные		
10:35 11:55	Вычислительные	Вычислительные	Вычислительные	Вычислительные		
12:25 13:45	Метрология И Физ.	Метрология И Физ.	Метрология И Физ.	Метрология И Физ.	Вычислительные	
14:00 15:20	Метрология И Физ.	Метрология И Физ.	Метрология И Физ.	Метрология И Физ.	Вычислительные	
15:50 17:10					Метрология И Физ.	
17:25 18:45					Метрология И Физ.	
19:00 20:20						
20:40 22:00						

Рисунок 3.3 – Подробная информация о сотруднике


КАФЕДРА
 Университета



Моя страница
 Состав
 История
 Документы

СОСТАВ КАФЕДРЫ

Алексеева Арина Дмитриевна
Доцент, канд. техн. наук

Балашова Варвара Львовна
Доцент, Д-р техн. наук

Беляева Мария Ярославовна
Профессор, Д-р техн. наук

Васильев Александр Николаевич
Профессор, Д-р физ.-мат. наук

Васильева Анна Артёмовна
Доцент, Д-р техн. наук

Васнев Прокофий Аркадьевич
Администратор, ассистент

Дернов Клаус Богуславович
Доцент, Д-р техн. наук



**Каверин Демьян
Зиновьевич**
 Доцент, канд. техн. наук
[подробнее](#)

Рисунок 3.4 – Интерфейс раздела «Состав кафедры»

Далее, при нажатии на кнопку «редактировать», пользователю предоставляется возможность редактировать следующие пункты: аудитория, телефон, e-mail. При редактировании поля «Консультации» изменения автоматически сохраняются.

В разделе «Расписание» отображается актуальное расписание преподавателя на неделю. Текущий день недели подсвечивается. При

наведении на ячейку отображаются название предмета, аудитория и группа, у которой проводится данное занятие.

В разделе «Состав» размещены данные о сотрудниках кафедры, а также информация о расположении кафедры. Интерфейс данного раздела показан на рисунке 3.4.

При нажатии на карточку с сотрудником появляется информация о нем. Результат показан на рисунке 3.5.

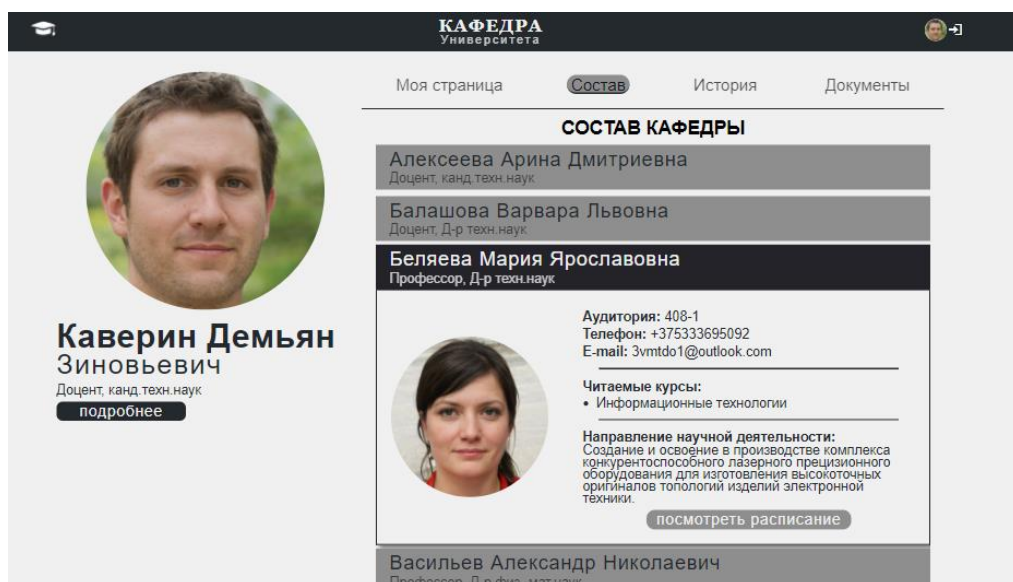


Рисунок 3.5 – Структура раздела «Состав кафедры»

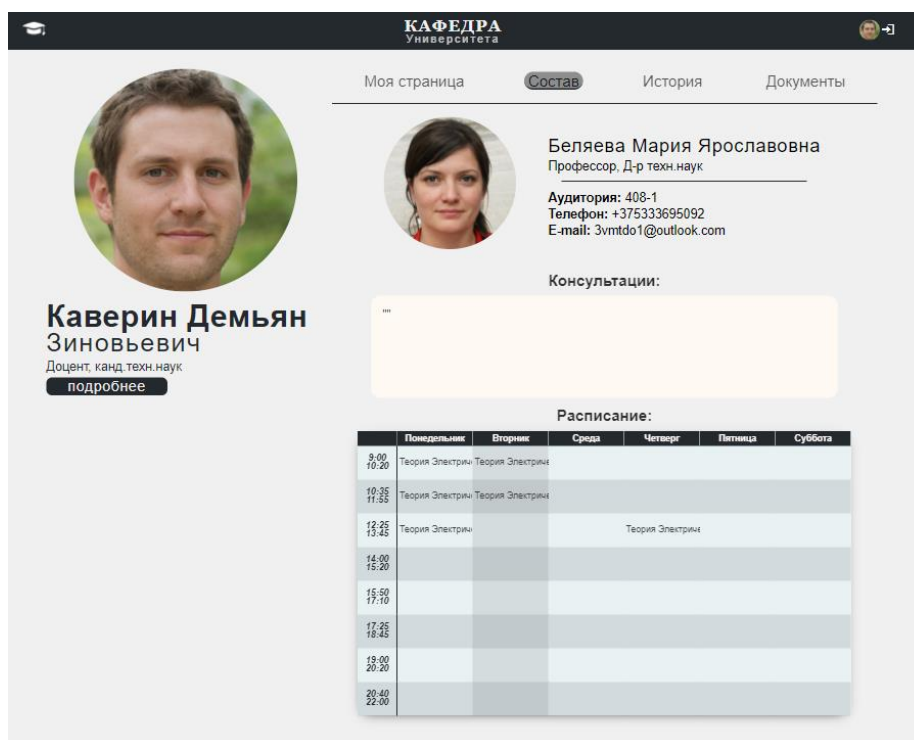


Рисунок 3.6 – Интерфейс страницы сотрудника

Далее, при нажатии на кнопку «посмотреть расписание», открывается страничка выбранного сотрудника. Результат показан на рисунке 3.6.

В разделе «История» отображена история кафедры. Интерфейс раздела показан на рисунке 3.7.

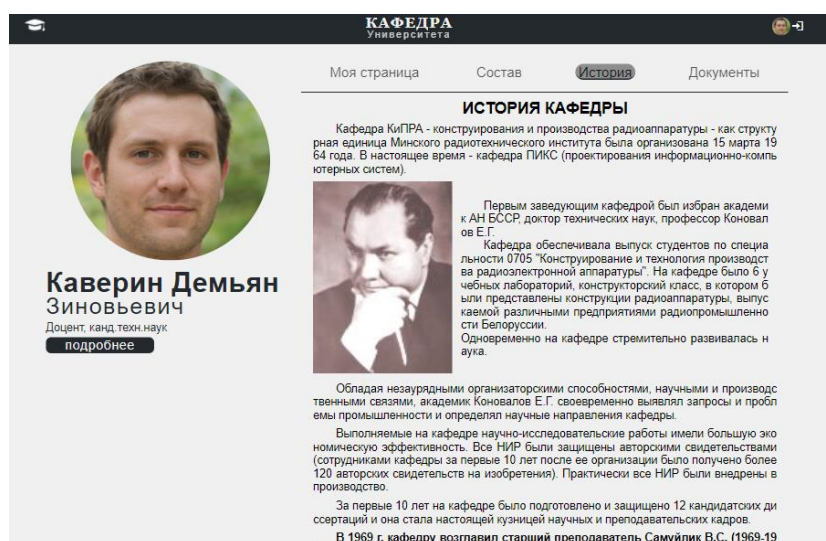


Рисунок 3.7 – Структура раздела «История кафедры»

При нажатии на раздел «Документы» происходит переход на страницу с документами кафедры.

В разделе «Помощь» размещены ответы на часто задаваемые вопросы, а также отображены контакты администратора. Интерфейс раздела изображен на рисунке 3.8.

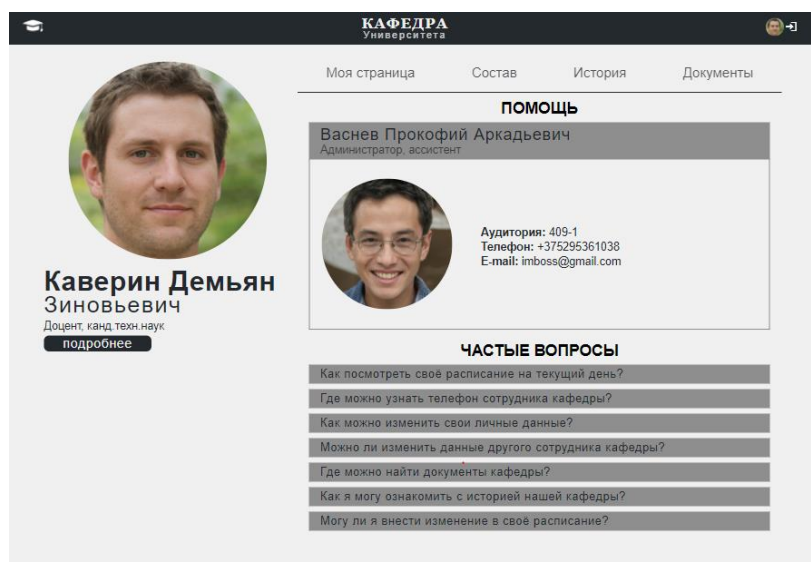


Рисунок 3.8 – Структура раздела «Помощь»

В разделе «Возможности» размещена информация о сайте и краткое руководство пользователя.

При нажатии на иконку выхода в правом верхнем углу происходит завершение сеанса и переадресация на страницу авторизации.

Интерфейс раздела «Состав кафедры» для администратора показан на рисунке 3.9.



Рисунок 3.9 – Структура раздела «Состав кафедры» для администратора

При нажатии на иконку «редактировать» (синего цвета), администратору предоставляется возможность редактировать данные сотрудника. Также появляется текстовое поле, в которое можно ввести новые логин и пароль пользователя.

При нажатии на красную кнопку – данные сотрудника удаляются из базы данных. При нажатии на зелёную кнопку появляется всплывающая форма, необходимая для добавления сотрудника в базу данных. Данная форма показана на рисунке 3.10.

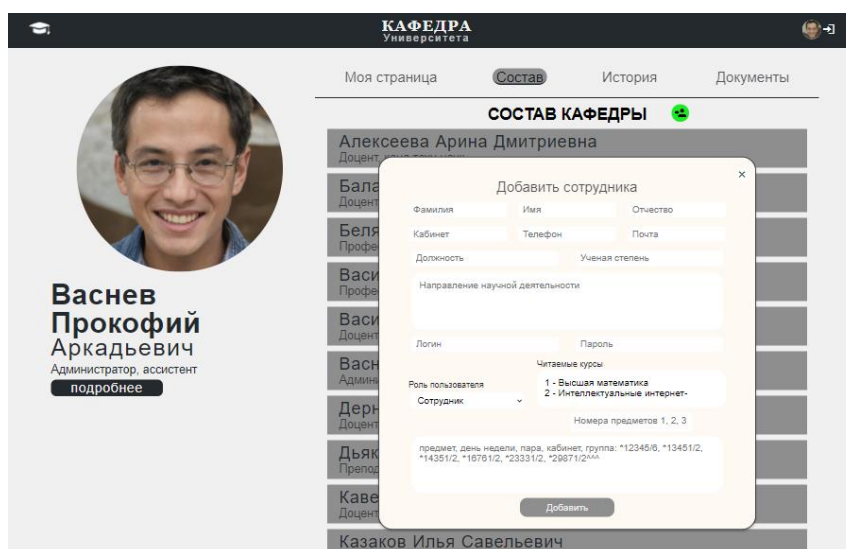


Рисунок 3.10 – Форма для добавления сотрудника в базу

Разработанное приложение является адаптивным, а значит одинаково корректно отображается на любых устройства. Интерфейс главной страницы приложения при просмотре со смартфона представлен на рисунке 3.11.

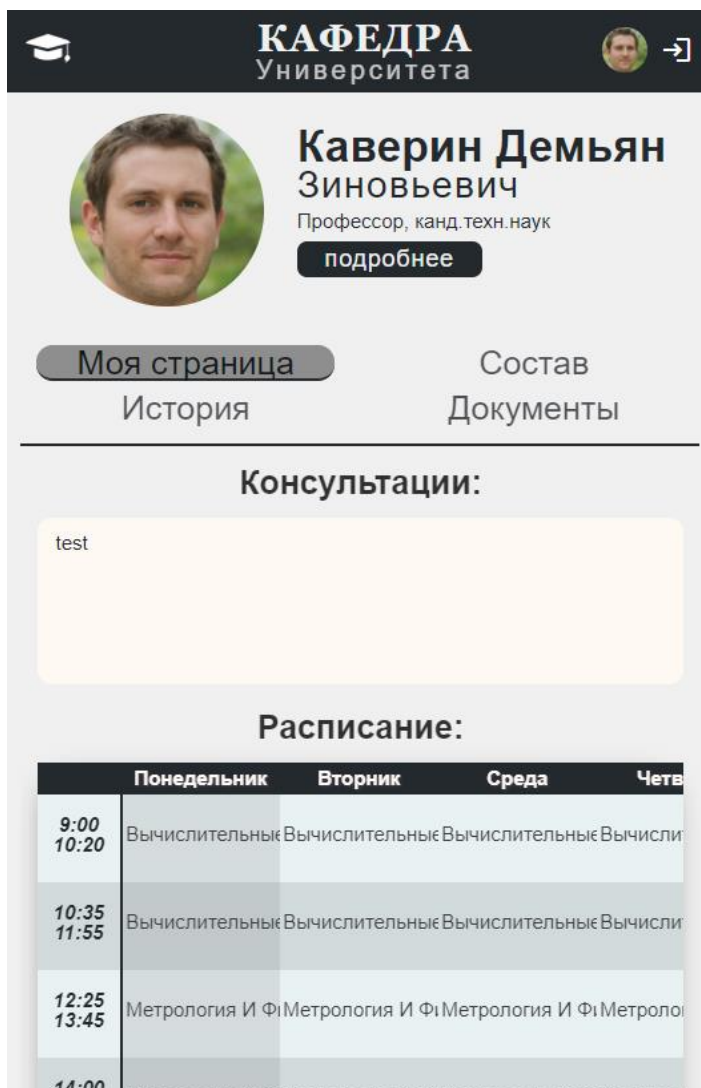


Рисунок 3.11 – Интерфейс приложения при просмотре со смартфона

3.2 Администрирование базы данных

В обязанности администратора MySQL входит создание и настройка учетных записей пользователей MySQL. В процессе этой настройки необходимо определить, какие пользователи будут иметь возможность подключения к серверу, откуда они смогут подключиться и что смогут делать после подключения.

Рассмотрим ключевые возможности пользователя с ролью «Сотрудник», реализованные в веб-приложении:

- авторизация пользователя;
- просмотр личной информации и информации о сотрудниках;
- редактирование личной информации;

- возможность просматривать расписание занятий сотрудников кафедры;
- возможность просматривать и скачивать документы кафедры;
- возможность публикации заметок и важной информации на личной странице.

Для этого данной группе пользователей должны быть доступны для просмотра сущности, «prepos_job», «courses», «subjects», «groups_stud», «timetable». Кроме того, сущности «prepos» и «consultation» должны быть доступны для просмотра и редактирования.

Создание пользователя «Сотрудник» и наделение его соответствующими правами приведено на рисунке 3.12.

```
CREATE USER 'super_user'@'localhost' IDENTIFIED BY 'super';
GRANT SELECT, UPDATE ON test_with_inc.prepos TO 'super_user'@'localhost';
GRANT SELECT ON test_with_inc.prepos_job TO 'super_user'@'localhost';
GRANT SELECT ON test_with_inc.courses TO 'super_user'@'localhost';
GRANT SELECT ON test_with_inc.subjects TO 'super_user'@'localhost';
GRANT SELECT ON test_with_inc.groups_stud TO 'super_user'@'localhost';
GRANT SELECT, UPDATE ON test_with_inc.consultation TO 'super_user'@'localhost';
GRANT SELECT ON test_with_inc.timetable TO 'super_user'@'localhost';
```

Рисунок 3.12 – Создание пользователя «Сотрудник»

Подключение пользователя «Сотрудник» к базе данных осуществляется следующим образом:

```
$auth = mysqli_connect('localhost', 'super_user', 'super',
'test_with_inc');
```

К важнейшим функциям, доступным пользователю с ролью «Администратор», относятся:

- создание и удаление пользователей;
- редактирование информации о сотрудниках;
- редактирование документов кафедры.

Создание пользователя «Администратор» и наделение его соответствующими правами осуществляется так:

```
CREATE USER 'db_admin'@'localhost' IDENTIFIED BY 'admin';
GRANT ALL ON test_with_inc.* TO 'db_admin'@'localhost';
```

Подключение пользователя «Администратор» к базе данных осуществляется следующим образом:

```
$auth = mysqli_connect('localhost', 'db_admin', 'admin',  
'test_with_inc');
```

Важным моментом в обеспечении сохранности данных является проверка созданных резервных копий – так называемые «учения», с полным разворачиванием базы данных из созданной резервной копии и последующим тестированием сохранности данных.

3.3 Реализация клиентских запросов

Организация доступа веб-сервера к ресурсам сервера базы данных была реализована с помощью технологии *OpenServer*. Программный комплекс имеет богатый набор серверного программного обеспечения, удобный, многофункциональный продуманный интерфейс, обладает мощными возможностями по администрированию и настройке компонентов. Платформа широко используется с целью разработки, отладки и тестирования веб-проектов, а также для предоставления веб-сервисов в локальных сетях.

Удобство и простота управления позволяет считать данный комплекс первоклассным и надёжным инструментом.

Использование технологии *OpenServer* позволяет разработчику комфортно себя чувствовать в процессе проектирования [9].

Чтобы редактировать личную информацию сотрудника в базе данных на сервер отправляются следующие запросы:

```
UPDATE `test_with_inc`.`prepods` SET last_name = '$newPersonData[0]'  
where user_id = '$thisPersonId';  
UPDATE `test_with_inc`.`prepods` SET first_name = '$newPersonData[1]'  
where user_id = '$thisPersonId';  
UPDATE `test_with_inc`.`prepods` SET fathers_name = '$newPersonData[2]'  
where user_id = '$thisPersonId';  
UPDATE `test_with_inc`.`prepods_job` SET position_name = '$newPerson-  
Data[3]' where job_id = '$thisPersonId';  
UPDATE `test_with_inc`.`prepods_job` SET ac_degree = '$newPersonData[4]'  
where job_id = '$thisPersonId';  
UPDATE `test_with_inc`.`prepods` SET location = '$newPersonData[5]' where  
user_id = '$thisPersonId';  
UPDATE `test_with_inc`.`prepods` SET phone = '$newPersonData[6]' where  
user_id = '$thisPersonId';  
UPDATE `test_with_inc`.`prepods` SET email = '$newPersonData[7]' where  
user_id = '$thisPersonId';  
UPDATE `test_with_inc`.`prepods_job` SET science_direction = '$newPerson-  
Data[8]' where job_id = '$thisPersonId';
```

В данных запросах переменная *\$newPersonData* представляет собой массив, содержащий данные, заполненные пользователем при редактировании.

Для просмотра информации о сотрудниках используется следующий запрос:

```
SELECT * FROM `test_with_inc`.`prepods` inner join
`test_with_inc`.`prepods_job` on job_id = user_id order by last_name asc;
```

Данный запрос выводит таблицу, содержащую личную информацию о всех сотрудниках кафедры, а также информацию о их деятельности, отсортированную по фамилии в алфавитном порядке. Для просмотра информации о читаемых курсах сотрудника используется следующий запрос:

```
SELECT subject_name FROM `test_with_inc`.`prepods` inner join
`test_with_inc`.`courses` on user_id = for_user_id inner join
`test_with_inc`.`subjects` on subject_name_id = subject_of_course_id WHERE
`user_id` = '$prepodsIdArray[$i]' ORDER BY subject_name DESC;
```

Данный запрос выводит таблицу, содержащую читаемые курсы преподавателя, отсортированные в обратном порядке следования букв алфавита. Для просмотра информации о расписании сотрудника используется следующий запрос:

```
SELECT subject_name, subject_day, subject_start, subject_location,
group_name_id, speciality_name FROM `test_with_inc`.`timetable` inner join
`test_with_inc`.`groups_stud` on group_stud_id = group_number inner join
`test_with_inc`.`subjects` on subject_id = subject_name_id WHERE for_prepod =
'$thisPersonConsultId' ORDER BY subject_day;
```

Данный запрос выводит таблицу, содержащую информацию о расписании преподавателя на неделю, отсортированную в порядке следования дней недели. Для внесения изменений в значение текстового поля в сущности «Консультации» используется следующий запрос:

```
UPDATE `test_with_inc`.`consultation` SET user_text_consult = '$newValue'
where consult_id = '$thisPersonConsultId';
```

В данном запросе переменная `$thisPersonConsultId` соответствует ID пользователя.

Листинги программного кода представлены в Приложении В.

Процесс разработки проекта отражен в GitHub репозитории <https://github.com/AndrewShamrey/course-project-Database-of-the-University-Department->.

3.4 Обоснование и реализация механизма обеспечения безопасности и сохранности данных

Для сохранности данных пользователя необходимо обеспечить базовые механизмы безопасности для веб-приложений. К ним относится, например, шифрование паролей пользователей системы для избежания

несанкционированного доступа и конфиденциальности информации даже в случае взлома таблицы с данными от аккаунта.

В качестве хэш-функции для зашифрованного хранения паролей в данном проекте была использована *md5*.

128-битный алгоритм хеширования, разработанный в 1991 году, предназначен для создания «отпечатков» или дайджестов сообщения произвольной длины и последующей проверки их подлинности. Широко применялся для проверки целостности информации и хранения хешей паролей.

На вход алгоритма поступает входной поток данных, хеш которого необходимо найти. Длина сообщения измеряется в битах и может быть любой (в том числе нулевой).

Этот способ шифрования является самым распространенным способом защитить информацию в сфере прикладных исследований, а также в области разработки веб-приложений. Хеш необходимо обезопасить от всевозможных хакерских атак [10].

Из вышеннаписанного можно сделать вывод, что любому программному продукту необходим механизм безопасности и сохранности данных, который позволит сохранить личные данные пользователей при попытке взлома и не даст злоумышленнику получить контроль над системой.

Для обеспечения сохранности данных выполняется резервное копирование данных. В *MySQL* это можно сделать, перейдя на вкладку «*Server*»—«*Data Export*». Восстановить базу данных по ранее созданному дампу можно перейдя на вкладку «*Server*»—«*Data Import*».

ЗАКЛЮЧЕНИЕ

В ходе разработки курсового проекта была разработана база данных для поддержания работы кафедры. Построена логическая и физическая модель базы данных, созданы и заполнены необходимые таблицы.

В ходе разработки курсового проекта были полностью выполнены следующие задачи:

- изучена предметная область и её информационные потребности;
- изучены такие языки программирования и разметки, как *HTML*, *CSS*, *JavaScript* и *PHP*, а также язык программирования структурированных запросов к базе данных *SQL*;
- приобретены навыки организации взаимодействия разработанных баз данных и ПО;
- изучены основные задачи администрирования и анализа баз данных;
- разработана база данных для поддержания работы кафедры;
- разработано веб-приложение, реализующее механизмы работы с данными в базе данных.

Каждая таблица спроектированной базы данных проверена на соответствие требованиям 3 нормальной форме для достижения минимальной избыточности данных.

В приложении предусмотрена защита пользовательских данных путём ввода логина и пароля, системы ролей. При запуске приложения появляется окно авторизации, в котором сотрудник вводит логин и пароль, в случае если данные верны, происходит заполнение страницы необходимыми данными.

Интерфейс системы спроектирован с учетом работы неопытных пользователей без необходимости дополнительного обучения, что позволяет уменьшить временные затраты на внедрение программного продукта. Реализованы следующие функции обработки данных: пополнение базы данных, изменение (редактирование и удаление) данных.

Разработанная в ходе написания курсовой работы информационная система упрощает работу и взаимодействие сотрудников кафедры, как в плане уменьшения затраченного времени, так и со стороны удобства получения необходимой информации. При использовании полученной системы пользователь, имеющий пароль для доступа в качестве администратора может изменять и удалять записи. Клиент в свою очередь имеет возможность просматривать необходимые для него таблицы, редактировать личную информацию.

Полученные в ходе выполнения данного курсового проекта навыки и умения, несомненно, будут полезны в будущем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Алексеев, В. Ф. Структуры и базы данных. Пособие для курсового проектирования / Минск : БГУИР, 2017.
- [2] Официальный сайт Белорусского Государственного экономического. Лекция «Реляционная модель данных» [Электронный ресурс]. – Режим доступа : http://www.bseu.by/it/tohod/lekcii2_3.htm.
- [3] Лукин, В. Н. Базы данных : конспект лекций / В. Н. Лукин – Москва : Московский авиационный институт, 2009. – 105 с.
- [4] Studfiles. Файловый архив студентов. Инфологическая модель данных «сущность-связь» [Электронный ресурс]. – Режим доступа : <https://studfiles.net/preview/720297/>.
- [5] Studfiles. Файловый архив студентов. Основы проектирования баз данных [Электронный ресурс]. – Режим доступа : <https://studfiles.net/preview/1587882/page:2/>.
- [6] web-creator.ru. Сложные IT-проекты. MySQL – система управления базами данных [Электронный ресурс]. – Режим доступа : <https://web-creator.ru/articles/mysql>.
- [7] Studfiles. Файловый архив студентов. Проектирование запросов [Электронный ресурс]. – Режим доступа : <https://studfiles.net/preview/5407090/page:16/>.
- [8] Основы работы с MySQL Workbench [Электронный ресурс]. – Режим доступа : <http://mithrandir.ru/professional/soft-and-hardware/mysql-workbench-basics.html>.
- [9] Официальный сайт Open Server Panel [Электронный ресурс]. – Режим доступа : <https://ospanel.io/>.
- [10] Шифрование MD5 [Электронный ресурс]. – Режим доступа : <https://cryptoperson.ru/cryptography/shifrovanie-md5>.

ПРИЛОЖЕНИЕ А

(обязательное)

Отчёт о проверке на уникальность в системе «Антиплагиат»

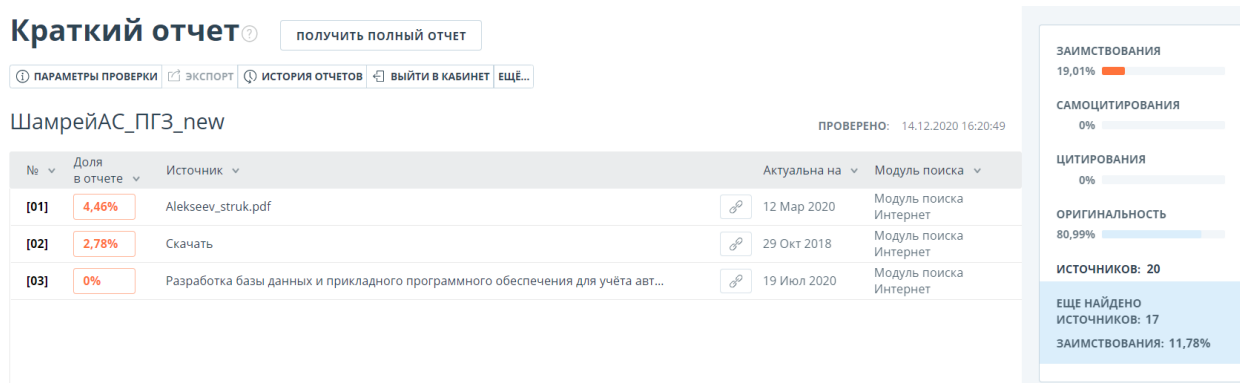


Рисунок А.1 – Отчёт о проверке на уникальность в системе «Антиплагиат»

ПРИЛОЖЕНИЕ Б

(обязательное)

Скрипт генерации базы данных

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----
-- Schema test_with_inc
-- -----

-- -----
-- Schema test_with_inc
-- -----
CREATE SCHEMA IF NOT EXISTS `test_with_inc` DEFAULT CHARACTER SET utf8 ;
-- -----
-- Schema my_project
-- -----
USE `test_with_inc` ;

-- -----
-- Table `test_with_inc`.`prepods`
-- -----
CREATE TABLE IF NOT EXISTS `test_with_inc`.`prepods` (
  `user_id` INT NOT NULL,
  `last_name` VARCHAR(45) NULL,
  `first_name` VARCHAR(45) NULL,
  `fathers_name` VARCHAR(50) NULL,
  `location` VARCHAR(10) NULL,
  `phone` VARCHAR(30) NULL,
  `email` VARCHAR(50) NULL,
  PRIMARY KEY (`user_id`))
ENGINE = InnoDB;

-- -----
-- Table `test_with_inc`.`authorization`
-- -----
CREATE TABLE IF NOT EXISTS `test_with_inc`.`authorization` (
  `auth_id` INT NOT NULL,
  `login` VARCHAR(45) NULL,
  `pass` CHAR(32) NULL,
  `role_id` TINYINT NOT NULL,
  PRIMARY KEY (`auth_id`),
  CONSTRAINT `auth_id`
    FOREIGN KEY (`auth_id`)
      REFERENCES `test_with_inc`.`prepods` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `test_with_inc`.`consultation`
-- -----
CREATE TABLE IF NOT EXISTS `test_with_inc`.`consultation` (
  `consult_id` INT NOT NULL,
```



```

    `user_text_consult` TEXT NULL,
    PRIMARY KEY (`consult_id`),
    UNIQUE INDEX `user_id_UNIQUE` (`consult_id` ASC) VISIBLE,
    CONSTRAINT `consult_id`
        FOREIGN KEY (`consult_id`)
        REFERENCES `test_with_inc`.`prepods` (`user_id`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `test_with_inc`.`subjects`
-- -----
CREATE TABLE IF NOT EXISTS `test_with_inc`.`subjects` (
  `subject_name_id` INT NOT NULL,
  `subject_name` TEXT NULL,
  PRIMARY KEY (`subject_name_id`))
ENGINE = InnoDB;

-- -----
-- Table `test_with_inc`.`courses`
-- -----
CREATE TABLE IF NOT EXISTS `test_with_inc`.`courses` (
  `user_subj_id` INT NOT NULL,
  `for_user_id` INT NOT NULL,
  `subject_of_course_id` INT NOT NULL,
  PRIMARY KEY (`user_subj_id`),
  INDEX `subject_of_course_id_idx` (`subject_of_course_id` ASC) VISIBLE,
  INDEX `for_user_id_idx` (`for_user_id` ASC) VISIBLE,
  CONSTRAINT `for_user_id`
    FOREIGN KEY (`for_user_id`)
    REFERENCES `test_with_inc`.`prepods` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `subject_of_course_id`
    FOREIGN KEY (`subject_of_course_id`)
    REFERENCES `test_with_inc`.`subjects` (`subject_name_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `test_with_inc`.`groups_stud`
-- -----
CREATE TABLE IF NOT EXISTS `test_with_inc`.`groups_stud` (
  `group_stud_id` INT NOT NULL,
  `group_name_id` INT NULL,
  `speciality_name` TEXT NULL,
  PRIMARY KEY (`group_stud_id`))
ENGINE = InnoDB;

-- -----
-- Table `test_with_inc`.`prepods_job`
-- -----
CREATE TABLE IF NOT EXISTS `test_with_inc`.`prepods_job` (
  `job_id` INT NOT NULL,
  `position_name` VARCHAR(45) NULL,
  `ac_degree` TEXT NULL,
  `science_direction` TEXT NULL,
  PRIMARY KEY (`job_id`),
  UNIQUE INDEX `user_id_UNIQUE` (`job_id` ASC) VISIBLE,
  CONSTRAINT `job_id`

```

```

        FOREIGN KEY (`job_id`)
        REFERENCES `test_with_inc`.`prepods` (`user_id`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `test_with_inc`.`timetable`
-- -----
CREATE TABLE IF NOT EXISTS `test_with_inc`.`timetable` (
  `table_id` INT NOT NULL,
  `for_prepod` INT NOT NULL,
  `subject_id` INT NOT NULL,
  `subject_start` INT NULL,
  `subject_day` INT NULL,
  `subject_location` VARCHAR(10) NULL,
  `group_number` INT NOT NULL,
  PRIMARY KEY (`table_id`),
  INDEX `subject_id_idx` (`subject_id` ASC) VISIBLE,
  INDEX `for_prepod_idx` (`for_prepod` ASC) VISIBLE,
  INDEX `group_number_idx` (`group_number` ASC) VISIBLE,
  CONSTRAINT `for_prepod`
    FOREIGN KEY (`for_prepod`)
    REFERENCES `test_with_inc`.`prepods` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `group_number`
    FOREIGN KEY (`group_number`)
    REFERENCES `test_with_inc`.`groups_stud` (`group_stud_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `subject_id`
    FOREIGN KEY (`subject_id`)
    REFERENCES `test_with_inc`.`subjects` (`subject_name_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

ПРИЛОЖЕНИЕ В

(Листинги программного кода)

Подключение к базе данных:

```
$auth = mysqli_connect('localhost', 'for_connect', 'password',  
'test_with_inc');  
if (!$auth) {die('Error connect to DataBase with logins');}
```

Создание суперглобальной переменной \$_SESSION:

```
$login = strtolower($_POST['login']);  
$password = md5($_POST['password']);  
$check_user = mysqli_query($auth, "SELECT * FROM `test_with_inc`.`authori-  
zation` WHERE `login` = '$login' AND `pass` = '$password'");  
if (mysqli_num_rows($check_user) > 0) {  
    $user = mysqli_fetch_assoc($check_user);  
    $_SESSION['user'] = [  
        "id" => $user['auth_id'],  
        "login" => $user['login'],  
        "role" => $user['role_id'],  
    ];  
    header('Location: main/index.php', true);  
} else {  
    $_SESSION['message'] = 'Неверный логин или пароль';  
    header('Location: authorization/connection.php');  
}
```

Завершение сеанса:

```
unset($_SESSION['user']);  
header('Location: authorization/connection.php');
```

Удаление пользователя:

```
$personToDrop = $_POST['delete_person'];  
$deletePersonQuery = mysqli_query($auth, "DELETE FROM  
`test_with_inc`.`prepods` WHERE user_id = '$personToDrop'");
```

Редактирование поля «Консультации»:

```
$newValue = nl2br($_POST['consultations']);  
$thisPersonConsultId = $_SESSION['user']['id'];  
$changeConsultData = mysqli_query($auth, "UPDATE `test_with_inc`.`consul-  
tation` SET user_text_consult = '$newValue' where consult_id = '$thisPer-  
sonConsultId'");
```

Редактирование пользователями личных данных:

```
$thisPersonId = $_SESSION['user']['id'];  
$newPersonData = explode(' ', $_POST['change_data']);  
$changePersonLocation = mysqli_query($auth, "UPDATE  
`test_with_inc`.`prepods` SET location = '$newPersonData[0]' where user_id =  
'$thisPersonId'");  
$changePersonPhone = mysqli_query($auth, "UPDATE `test_with_inc`.`prepods`  
SET phone = '$newPersonData[1]' where user_id = '$thisPersonId'");
```

```
$changePersonEmail = mysqli_query($auth, "UPDATE `test_with_inc`.`prepods` SET email = '$newPersonData[2]' where user_id = '$thisPersonId'");
```

Редактирование данных администратором:

```
$changePersonLastName = mysqli_query($auth, "UPDATE `test_with_inc`.`prepods` SET last_name = '$newPersonData[0]' where user_id = '$thisPersonId'");
$changePersonFirstName = mysqli_query($auth, "UPDATE `test_with_inc`.`prepods` SET first_name = '$newPersonData[1]' where user_id = '$thisPersonId'");
$changePersonFathersName = mysqli_query($auth, "UPDATE `test_with_inc`.`prepods` SET fathers_name = '$newPersonData[2]' where user_id = '$thisPersonId'");
$changePersonPositionName = mysqli_query($auth, "UPDATE `test_with_inc`.`prepods_job` SET position_name = '$newPersonData[3]' where job_id = '$thisPersonId'");
$changePersonAcDegree = mysqli_query($auth, "UPDATE `test_with_inc`.`prepods_job` SET ac_degree = '$newPersonData[4]' where job_id = '$thisPersonId'");
$changePersonLocation = mysqli_query($auth, "UPDATE `test_with_inc`.`prepods` SET location = '$newPersonData[5]' where user_id = '$thisPersonId'");
$changePersonPhone = mysqli_query($auth, "UPDATE `test_with_inc`.`prepods` SET phone = '$newPersonData[6]' where user_id = '$thisPersonId'");
$changePersonEmail = mysqli_query($auth, "UPDATE `test_with_inc`.`prepods` SET email = '$newPersonData[7]' where user_id = '$thisPersonId'");
$changePersonScience = mysqli_query($auth, "UPDATE `test_with_inc`.`prepods_job` SET science_direction = '$newPersonData[8]' where job_id = '$thisPersonId'");
```

Создание нового пользователя:

```
$lastName = $_POST['last_name'];
$firstName = $_POST['first_name'];
$fathersName = $_POST['fathers_name'];
$location = $_POST['location'];
$phone = $_POST['phone'];
$email = $_POST['email'];
$positionName = $_POST['position_name'];
$acDegree = $_POST['ac_degree'];
$scienceDir = nl2br($_POST['science_direction']);
$login = strtolower($_POST['login']);
$password = md5($_POST['pass']);
$role = $_POST['role_id'];
$courses = nl2br($_POST['subj_of_course_id']);
$subjects = nl2br($_POST['timetable_data']);
$IdQuery = mysqli_query($auth, "SELECT MAX(user_id) FROM `test_with_inc`.`prepods`");
$newId = mysqli_fetch_assoc($IdQuery)['MAX(user_id)'] + 1;

$createUser = mysqli_query($auth, "INSERT INTO `test_with_inc`.`prepods` VALUES ('$newId', '$lastName', '$firstName', '$fathersName', '$location', '$phone', '$email')");
$createUserJob = mysqli_query($auth, "INSERT INTO `test_with_inc`.`prepods_job` VALUES ('$newId', '$positionName', '$acDegree', '$scienceDir')");
$createUserAuth = mysqli_query($auth, "INSERT INTO `test_with_inc`.`authorization` VALUES ('$newId', '$login', '$password', '$role')");
$createUserConsult = mysqli_query($auth, "INSERT INTO `test_with_inc`.`consultation` VALUES ('$newId', '')");
```

```

    $coursesArr = explode(',', $courses);
    for ($i = 0 ; $i < count($coursesArr) ; $i++) {
        $currentCourseId = mysqli_fetch_assoc(mysqli_query($auth, "SELECT
        MAX(user_subj_id) FROM `test_with_inc`.`courses`"))['MAX(user_subj_id)'] + 1;
        $createUserAuth = mysqli_query($auth, "INSERT INTO
        `test_with_inc`.`courses` VALUES ('$currentCourseId', '$newId',
        '$coursesArr[$i]')");
    }

    $subjectsArr = explode('^^^', $subjects);
    for ($i = 0 ; $i < count($subjectsArr) ; $i++) {
        $currentTimeTableSubj = explode(',', $subjectsArr[$i]);
        $currentTimeTableId = mysqli_fetch_assoc(mysqli_query($auth, "SELECT
        MAX(table_id) FROM `test_with_inc`.`timetable`"))['MAX(table_id)'] + 1;
        $queryToGroups = mysqli_query($auth, "SELECT group_stud_id FROM
        `test_with_inc`.`groups_stud` WHERE group_name_id = $currentTimeTable-
        Subj[4]");
        $queryToGroupsReady = mysqli_fetch_assoc($que-
        ryToGroups)['group_stud_id'];
        $currentTimeTableQuery = mysqli_query($auth, "INSERT INTO
        `test_with_inc`.`timetable` VALUES ('$currentTimeTableId', '$newId', '$cur-
        rentTimeTableSubj[0]', '$currentTimeTableSubj[2]', '$currentTimeTable-
        Subj[1]', '$currentTimeTableSubj[3]', '$queryToGroupsReady')");}

```

Получение данных администратора:

```

    $adminQuery = mysqli_query($auth, "SELECT * FROM `test_with_inc`.`prepods`
    inner join `test_with_inc`.`prepods_job` on user_id = job_id inner join
    `test_with_inc`.`authorization` on auth_id = user_id where role_id = 2");
    $adminData = mysqli_fetch_assoc($adminQuery);
    $admin_data = [$adminData['user_id'], $adminData['last_name'], $ad-
    minData['first_name'], $adminData['fathers_name'], $adminData['location'],
    $adminData['phone'], $adminData['email'], $adminData['position_name'], $ad-
    minData['ac_degree']];
    $admin_data_str = implode(';;', $admin_data);

```