

Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

**ИЗУЧЕНИЕ ОСНОВ ЯЗЫКА РЕЛЯЦИОННЫХ БАЗ ДАННЫХ
- SQL**

КУРСОВАЯ РАБОТА

студента 2 курса 251 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Рогова Максима Игоревича

Научный руководитель
Старший преподаватель _____ М. И. Сафончик

Заведующий кафедрой
к. ф.-м. н. _____ С. В. Миронов

Саратов 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Основные понятия баз данных	4
1.1 Реляционные базы данных	4
2 Этапы проектирования реляционной базы данных	8
2.1 Анализ требований и проектирование	8
2.2 Модель сущность - связь	8
2.3 Нормализация	9
2.4 Физическая модель	10
2.5 Архитектура СУБД	10
3 Способы манипулирования данными с помощью языка SQL	13
4 Реализация базы данных поиска студентов	14
4.1 Описание интерфейса	14
4.2 Пользовательский сценарий	14
4.2.1 Главная страница	14
4.2.2 Страница входа	15
4.2.3 Страница восстановления пароля	15
4.2.4 Страница регистрации учетной записи	16
4.2.5 Страница результата для зарегистрированного пользователя	16
5 Реализация базы данных и подключение к WEB интерфейсу	21
5.1 Реализация реляционной базы данных на языке - SQL	21
5.2 Подключение базы данных к WEB интерфейсу	24
5.2.1 Подключение	25
5.2.2 Сессия	25
5.2.3 Запросы к базе данных	26
ЗАКЛЮЧЕНИЕ	28
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	29
Приложение А Список команд для реализации базы данных	30
Приложение Б Программный код для подключения базы данных к WEB интерфейсу	36
Приложение В ER - диаграмма	52

ВВЕДЕНИЕ

В современном мире приходится сталкиваться с огромным количеством информации, которую необходимо запоминать или где-то хранить. Так как человеческий мозг не может справиться с такой задачей на помощь приходят компьютеры, где самая разнообразная информация может храниться в разных форматах. Совокупность сведений о каких-либо объектах, процессах, событиях или явлениях, организованная таким образом, чтобы можно было легко представить любую часть этой совокупности, называют базой данных. Задачи хранения, получения, анализа данных принято называть управлением данными, а программы для решения подобных задач – системами управления базами данных (СУБД). Данная курсовая работа посвящена реляционным базам данных. "Реляционный Relation - обозначает взаимосвязанный. Реляционная база данных - это набор двумерных простых таблиц. Так как на сегодняшний день практически все работающие базы данных соответствуют реляционной модели, целью данной работы является изучение основ реляционных баз данных и языка манипулирования ими - SQL.

Поставлены задачи:

- изучить и описать основные понятия реляционных баз данных;
- подробно ознакомиться с этапами проектирования и разработки РБД;
- изучить основы языка манипулирования данными SQL;
- в качестве практической задачи, создать реляционную базу и реализовать пользовательский интерфейс с рядом типовых запросов на SQL.

Используемые инструменты и языки:

- язык SQL в среде MySql [1] для реализации реляционной базы данных;
- HTML5, CSS3(SASS), NODE.JS (Gulp + plugins), Bootstrap 4, JQuery для создания интерфейса базы данных;
- Node.js(express, mysql) для подключения СУБД к интерфейсу.

1 Основные понятия баз данных

В теории реляционную базу данных можно создать, не прибегая к помощи специальных инструментов. На практике при разработке реляционных баз данных используют средства систем управления базами данных (СУБД). СУБД иногда называют реляционными СУБД. (РСУБД) Реляционная база данных — это реализация реляционной модели (модели данных) на физическом уровне, и потому важно четко различать эти два понятия: модель данных и базу данных. Как правило, на стадии проектирования невозможно полностью изолироваться от ограничений, налагаемых средой разработки, в то время как в основу проекта рекомендуется закладывать максимально «чистую» модель. [2]

1.1 Реляционные базы данных

Реляционная база данных создана была для организации данных в таблицы и обеспечения операций извлечения, генерирующих новые таблицы из уже имеющихся. Основной областью, в которой произошло наиболее быстрое развитие баз данных, являлись разработки приложений для Интернета. База данных сервера поддерживает многие важные функции в Интернете. Фактически, любое содержание web - страниц может управляться базой данных.

По мере увеличения возможностей и уменьшения стоимости вычислительных средств, получило развитие второе направление, связанное с использованием средств вычислительной техники в автоматизированных информационных системах. Здесь вычислительные возможности компьютеров отходят на второй план - основные функции вычислительных средств в информационных системах состоят в поддержке надежного хранения информации, выполнении специфических для данного приложения преобразований информации и/или вычислений, предоставлении пользователям удобного и легко осваиваемого интерфейса. [3]

Особенности реляционных таблиц:

- Таблица имеет уникальное имя и состоит из однотипных строк;
- Существование фиксированного числа полей и значений;
- Возможность однозначной идентификации любой строки таблицы.

Основные понятия реляционных баз данных:

- Домен;
- Кортеж;
- Первичный ключ;
- Внешний ключ.

Домен - это семантическое понятие. Домен можно рассматривать как подмножество значений некоторого типа данных имеющих определенный смысл. Домен характеризуется следующими свойствами: имеет уникальное имя (в пределах базы данных). Домен определен на некотором простом типе данных или на другом домене. Домен может иметь некоторое логическое условие, позволяющее описать подмножество данных, допустимых для данного домена. Домен несет определенную смысловую нагрузку. Например, домен , имеющий смысл "возраст сотрудника" можно описать как следующее подмножество множества натуральных чисел: если тип данных можно считать множеством всех возможных значений данного типа, то домен напоминает подмножество в этом множестве. Отличие домена от понятия подмножества состоит именно в том, что домен отражает семантику, определенную предметной областью. Может быть несколько доменов, совпадающих как подмножества, но несущие различный смысл. Например, домены "Вес детали" и "Имеющееся количество" можно одинаково описать как множество неотрицательных целых чисел, но смысл этих доменов будет различным, и это будут различные домены. Основное значение доменов состоит в том, что домены ограничивают сравнения. Некорректно, с логической точки зрения, сравнивать значения из различных доменов, даже если они имеют одинаковый тип. В этом проявляется смысловое ограничение доменов. Синтаксически правильный запрос "выдать список всех деталей, у которых вес детали больше имеющегося количества" не соответствует смыслу понятий "количество" и "вес". Теперь более подробно рассмотрим основные типы данных языка SQL на рисунках 1, 2, 3, 4

CHAR (size)	Используется для хранения строк. Параметр в скобках позволяет фиксировать длину хранимой строки. Максимальный размер в байтах, который можно задать для строки, – 255.
VARCHAR (size)	Аналогично предыдущему типу позволяет хранить строки длиной до 255 символов. Однако отличие от CHAR в том, что для хранения значения данного типа выделяется требуемое количество памяти. То есть для строки, состоящей из 5-ти символов, потребуется 6 байт памяти. В первом же случае память для значения будет выделена согласно указанному параметру.
TINY TEXT	Используется для хранения строковых данных размером до 255 символов.
TEXT	Используется для хранения текстовой информации, размер которой не превышает 65 535 букв.
BLOB	Рассматриваемый тип данных аналогичен типу TEXT и позволяет сохранять в базе текстовую информацию, объем которой может достигать 65 535 знаков. Но на практике используется для хранения звуковых данных, рисунков, электронной документации и пр.
MEDIUM TEXT	Был разработан на базе типа TEXT, но позволяет хранить больше данных за счет увеличенного размера до 16 777 215 букв или символов.
MEDIUM BLOB	Используется для сохранения в базе электронных документов, размер которых не превышает отметку в 16 777 215 знаков.
LONG TEXT	Функционально аналогичен предыдущим типам, но с увеличенным объемом памяти до 4 гигабайт.
LONG BLOB	Позволяет помещать в базу данные больших объемов (4 294 967 295 символа).
ENUM (a, b, c, etc.)	Специальный тип данных, использующийся для задания списка возможных значений. Позволяет указать 65 535 значений. Строки рассматриваемого типа могут принимать единственное значение из указанных в множестве. В случае, когда будет происходить добавление значений, которые не присутствуют в заданном списке, в таблицу будут записаны пустые значения.

Рисунок 1 – Строковый тип данных

FLOAT (size, d)	Позволяет содержать дробные числа указываемой точности d.
DOUBLE (size, d)	Используется для хранения дробных чисел с двоичной точностью.
DECIMAL(size, d)	Хранение дробных значений в виде строк.

Рисунок 2 – Дробный тип данных

INT (size)	Хранение целочисленных значений, образующих диапазон [-2 ³¹ ; 2 ³¹ -1]
TINYINT (size)	Служит для хранения чисел в диапазоне от -128 до 127
SMALLINT (size)	Характеризуется увеличенным диапазоном хранимых значений в размере от -32 768 до 32 767
MEDIUMINT (size)	Используется для хранения чисел размерностью от -2 ²³ до 2 ²³ -1
BIGINT (size)	Охватывает диапазон целочисленных значений, начиная с -2 ⁶³ и заканчивая 2 ⁶³ -1

Рисунок 3 – Целочисленный тип данных

DATE	Главное предназначение - хранение даты в формате ГОД-МЕСЯЦ-ДЕНЬ ("ГГГГ-ММ-ДД" или "уууу-мм-дд"). Обычно значения разделены через «-», однако в качестве разделителя может быть задействован любой символ, кроме цифр.
TIME	Позволяет заносить в ячейку таблицы временные значения. Все значения задаются форматом «hh:mm:ss»
DATETIME	Объединяет функции предыдущих двух типов. Формат хранения представлен следующим образом: «уууу-мм-дд hh:mm:ss».
TIMESTAMP	Сохраняет дату и время, исчисляемое количеством секунд, прошедших начиная с полуночи 1.01.1970 года и до заданного значения.
YEAR (M)	Используется для хранения годовых значений в двух- или четырехзначном формате.

Рисунок 4 – Тип данных даты и времени

Кортеж - это группа взаимосвязанных элементов данных или по-другому строка таблицы реляционной базы данных.

Первичный ключ - в реляционной модели данных один из потенциальных ключей отношения, выбранный в качестве основного ключа. Первичный ключ: набор определенных признаков, уникальных для каждой записи. Обозначается первичный ключ, как primary key. Не позволяет создавать одинаковых записей (строк) в таблице и обеспечивают логическую связь между таблицами одной базы данных (для реляционных БД).

Внешний ключ - обеспечивает однозначную логическую связь, между таблицами одной БД. Например, есть две таблицы А и В. В таблице А (обувь), есть первичный ключ: размер, в таблице В (цвет) должна быть колонка с названием размер. В этой таблице «размер» это и будет внешний ключ для логической связи таблиц В и А.

2 Этапы проектирования реляционной базы данных

Разработка эффективной базы данных состоит из нескольких этапов. Процесс разработки БД начинается с анализа требований. Проектировщик на этом этапе разработки должен найти ответы на следующие вопросы: какие элементы данных должны храниться, кто и как будет к ним обращаться.

На втором этапе создается логическая структура БД. Для этого определяют, как данные будут сгруппированы логически. Структура БД на этом этапе выражается в терминах прикладных объектов и отношений между ними.

На заключительном (третьем) этапе логическая структура БД преобразуется в физическую с учетом аспектов производительности. Элементы данных на этом этапе получают атрибуты и определяются как столбцы в таблицах выбранной для реализации БД СУБД.

2.1 Анализ требований и проектирование

Требования к приложению с БД обычно составляются с помощью опросов и бесед с конечными пользователями. Это - итерационный процесс, в ходе которого разработчики определяют структуру пользовательских диалогов, критерии поиска документов и возможные реакции пользователей. Общая методика определения и документирования требований к БД заключается в составлении словаря данных. Словарь данных перечисляет и определяет отдельные элементы данных, которые должны храниться в базе. Собирает все необходимые данные в для реализации проекта.

Так же составляются ER - диаграммы, то есть связи данных в разных таблицах на основе каких-то ключевых параметров или по-другому ключей, отношения между объектами, которые могут быть сформулированы для простоты моделирования, пример: "Каждый студент учится в вузе, в каждом вузе есть кафедры, у каждой кафедры есть хотя бы одно направление."

2.2 Модель сущность - связь

Сущность — это нечто такое, о чем нужно хранить информацию в разрабатываемой системе. На концептуальном уровне связи представляют собой простые ассоциации между сущностями. Например, утверждение "Покупатели покупают продукты" указывает, что между сущностями Customers(Покупатели) и Products(Продукты) существует связь. В 1976 г. Питер Пин Шань Чен

(Peter Pin Shan Chen) внес существенный вклад в теорию моделирования данных, разработав модель «сущности — связи», в которой реализовано описание данных в терминах сущностей, атрибутов и связей. Одновременно он предложил новый метод построения диаграммы «сущности — связи» (Entity Relationship diagrams, или E/R diagrams), который вскоре стал широко применяться разработчиками баз данных. На диаграммах «сущности — связи», сущности изображаются в виде прямоугольников, атрибуты — эллипсов, а отношения — ромбов (см. рисунок 5). ER-диаграмма для данного предметной области в приложение B

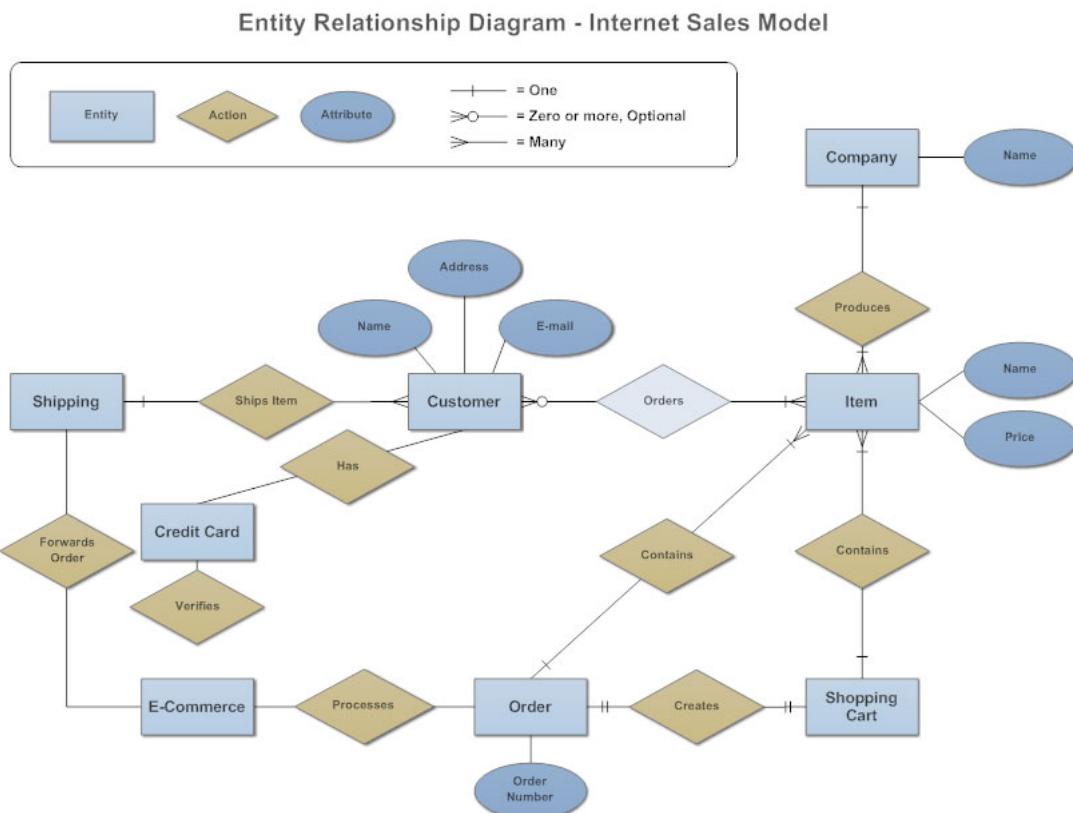


Рисунок 5 – E/R diagram

2.3 Нормализация

Далее все собранные данные и получившиеся таблицы нужно привести к нормальному виду, для соблюдения архитектуры нашей реляционной базы данных. Процесс нормализации состоит в пошаговом построении БД в нормальной форме, рассмотрим 4 нормальных формы реляционных баз данных:

- Первая нормальная форма (1НФ) очень проста. Все таблицы БД должны удовлетворять единственному требованию - каждая ячейка в табли-

цах должна содержать атомарное значение, другими словами, хранимое значение в рамках предметной области приложения БД не должно иметь внутренней структуры, элементы которой могут потребоваться приложению.

- Вторая нормальная форма (2НФ) создается тогда, когда удалены все частичные зависимости из отношений БД. Если в отношениях не имеется никаких составных ключей, то этот уровень нормализации легко достигается.
- Третья нормальная форма (3НФ) БД требует удаления всех транзитивных зависимостей.
- Четвертая нормальная форма (4НФ) создается при удалении всех многозначных зависимостей.

2.4 Физическая модель

Физическая модель – логическая модель базы данных, выраженная в терминах языка описания данных конкретной СУБД. Физическая модель базы данных содержит все детали, необходимые конкретной СУБД для создания базы: наименования таблиц и столбцов, типы данных, определения первичных и внешних ключей. Реляционная модель данных требует преобразования связей «многие ко многим» и иерархии наследования, и допускает их только на уровне логической модели базы данных.

2.5 Архитектура СУБД

Проблема отделения логической структуры баз данных и управление данными всталась давно, и решение этих проблем является развитием СУБД. Грубо говоря, пользовательский интерфейс СУБД должен быть отделен от физического интерфейса баз данных или физического представления баз данных, которое требуется компьютеру. Одна и та же база данных, в зависимости от того, как на нее смотреть может быть представлена по-разному, по-другому, база данных может иметь различные уровни описания и MySQL сервер в данном случае не является исключением. Число уровней описания зависит от реализации сервера баз данных или от реализации СУБД. Существует одноуровневая система управления базами данных. Двухуровневая СУБД и трехуровневая СУБД. Рассматривать одноуровневые и двухуровневые СУБД нет смысла, поскольку в настоящее время используются в основ-

ном трехуровневые системы. Системы с трехуровневым описанием данных имеют три уровня абстракции, на которых можно осуществляется взаимодействие с базой данных. Самым важным аспектом трехуровневой архитектуры базы данных является то, что логическая структура, с которой взаимодействует пользователь, отделена от физической структуры баз данных, с которой взаимодействует машина. См. рисунок 6

- В базах данных с трехуровневой архитектурой всегда есть внешний уровень, на котором пользователю предоставляется интерфейс для манипуляции данных, на внешнем уровне задаются права доступа пользователей на взаимодействие с базой данных, приоритеты и т.д.
- У трехуровневых СУБД есть всегда имеется внутренний уровень, на котором СУБД и операционная система воспринимают и обрабатывают данные
- Поскольку архитектура трехуровневая, то третий уровень отвечает за связь между внешним уровнем (пользовательским интерфейсом) и внутренним уровнем (физическими представлениями данных в базе данных). Третий уровень получил название концептуальный уровень представления данных. Концептуальный уровень представления данных предназначен для отображения внешнего уровня на внутренний, он обеспечивает независимость между этими уровнями. [4]

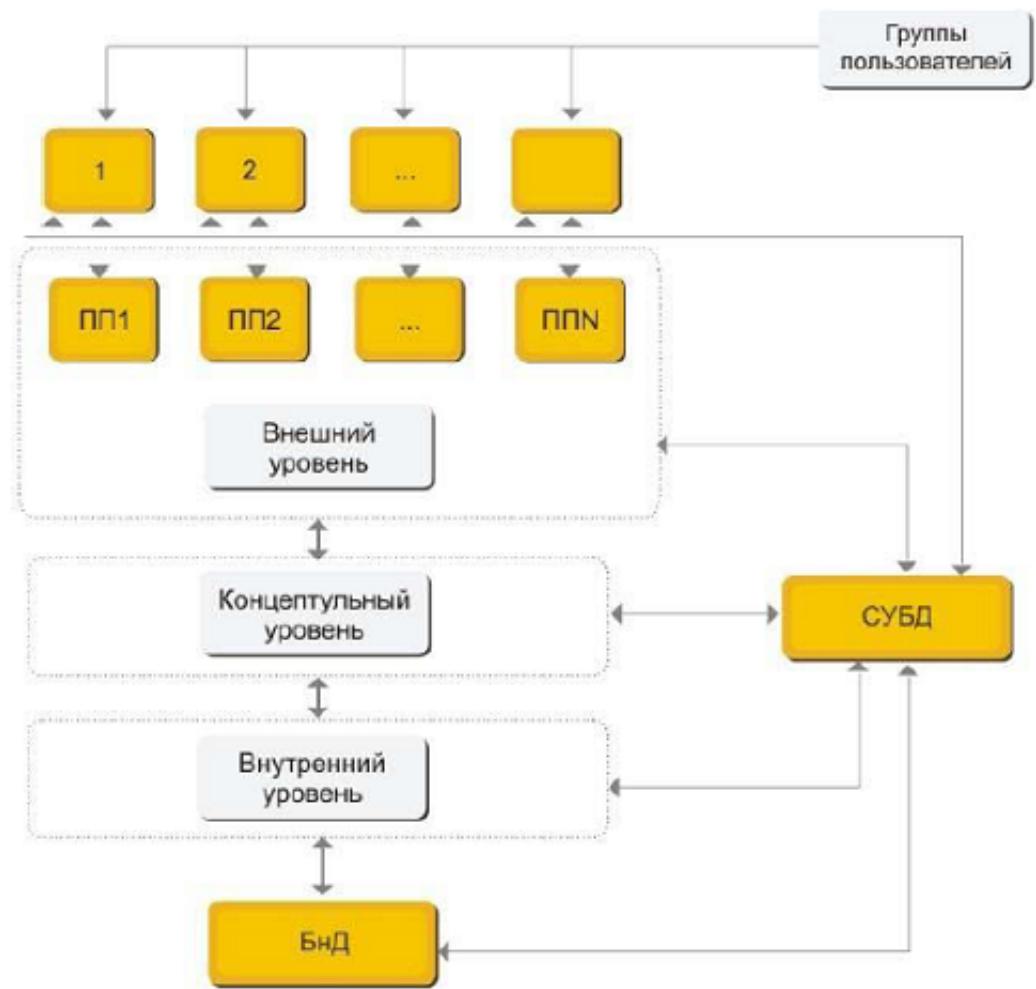


Рисунок 6 – Архитектура СУБД

3 Способы манипулирования данными с помощью языка SQL

Любой язык работы с базами данных должен предоставлять пользователю следующие возможности: создавать базы данных и таблицы с полным описанием их структуры; выполнять основные операции манипулирования данными, такие как вставка, модификация удаление данных из таблиц; выполнять простые и сложные запросы. Кроме того, язык работы с базами данных должен решать все указанные выше задачи при минимальных усилиях со стороны пользователя, а структура синтаксиса его команд должны быть достаточно просты и доступны для изучения. И, наконец, он должен быть универсальным, т.е. отвечать некоторому признанному стандарту, что позволит использовать один и тот же синтаксис и структуру команд при переходе от одной СУБД к другой. Язык SQL удовлетворяет практически всем этим требованиям. [5] Итак язык SQL состоит из 5-основных операторов:

- CREATE - создание, пример: CREATE table user;(что создаст таблицу user внутри базы данных)
- SELECT - вывод, пример: SELECT * FROM *username*;(выведет все значения из *username*)
- INSERT - добавление, пример: INSERT INTO user("Колонка1 "Колонка2 "Колонка3") (добавит в таблицу user с 3-мя колонками значения "Колонка1 "Колонка2 "Колонка3")
- UPDATE - обновление, пример: UPDATE S SET *STATUS* = *STATUS**2 WHERE *CITY* ='Paris'(удваивает статус всех поставщиков в Париже)
- DELETE - удаление, пример: DELETE FROM user WHERE *username* = 'Вася'(удаляет из таблицы user все строки, где значения колонки *username* = 'Вася') [6]

Более подробно рассмотрим язык SQL уже в главе 5.1

4 Реализация базы данных поиска студентов

4.1 Описание интерфейса

- Поиск студентами контактных данных других студентов с их факультета, с возможностью поиска по курсу, форме обучения, направлению;
- Получение контактных данных других студентов для помощи в учебе (сугубо по своим факультетам);
- Поиск студентов работодателями: по вузам, факультетам, направлениям, курс, формам обучения.

Пояснение к целям проекта:

- Студент может видеть других студентов, только по указанным им данных при регистрации о месте учебы, и не может видеть работодателей;
- Работодатель может видеть всех зарегистрированных студентов, но не может просматривать других работодателей.

4.2 Пользовательский сценарий

4.2.1 Главная страница

Пользователь попадает на главную страницу, на которой находится общая информация о ресурсе, его преимущества и цели. Цель страницы принудить к регистрации нового пользователя, или войти в учетную запись уже существующего пользователя - кнопка вход , регистрация и выход , найти людей (в случае активной сессии).

При нажатии кнопки Вход, пользователь будет перенаправлен на страницу входа.

В случае, если пользователь будет в текущей сессии на главной странице, все кнопки “Вход” на главной странице будут трансформированы в кнопку “Выход”, которая позволяет завершить текущую сессию.

Так же в случае активной сессии на главной странице, будет появляться кнопка “ Найти людей”, которая будет перенаправлять на страницу выдачи.

При нажатии кнопки Регистрация, пользователь будет перенаправлен на страницу регистрации.

4.2.2 Страница входа

На странице входа - user вводит email и пароль, после чего нажимает кнопку вход, которая перенаправляет его на страницу выдачи информации в соответствие с данными учетной записи.

На странице входа будет кнопка “Забыли пароль”, которая будет перенаправлять пользователя на отдельную страницу восстановления пароля.

Так же будет кнопка “Регистрация”, которая перенаправляет пользователя на страницу регистрации.

И кнопка главная, которая перенаправляет пользователя на главную страницу.

4.2.3 Страница восстановления пароля

На странице восстановления пароля при вводе email и нажатия кнопки Восстановить пароль, при наличие такого в бд учетных данных, будет отправлено письмо с паролем, принадлежащим к данному email и всплывающее окно "пароль отправлен на вашу почту по нажатию “Ок” будет перенаправлен на страницу Вход , иначе ошибка об отсутствие пользователя с этим email, и пользователь остается на этой же странице.

Так же будет кнопка “Регистрация”, которая перенаправляет пользователя на страницу регистрации.

И кнопка главная, которая перенаправляет пользователя на главную страницу.

Кнопка вход, которая будет перенаправлять на страницу входа.

4.2.4 Страница регистрации учетной записи

На странице регистрации у пользователя есть возможность вернуться на главную страницу, так же есть кнопки, переключающие регистрацию для Студента и Работодателя(рорип).

Для вкладки студент обязательными полями будут являться: Фамилия, Имя, Е-mail, Вуз, Курс, Факультет, Форма обучения, Направление, галочка согласия с обработкой персональных данных. Дополнительными полями являются: Отчество и дополнительные контактные данные.

Для вкладки Работодатель обязательными полями будут: Фамилия, Имя, Email, галочка согласия на обработку персональных данных. Дополнительные поля: Отчество, дополнительные контактные данные.

На каждой из форм регистрации будет кнопка “Политика обработки персональных данных” , при нажатии на которую будет всплывать модальное окно с текстом.

На каждой из форм регистрации будет кнопка “Зарегистрироваться” , при нажатие на которую в случае успешного заполнения обязательных полей, пользователь будет перенаправлен на страницу входа, иначе будет отображена текущая страница с ошибкой у соответствующего не правильно заполненного поля.

Для полей вуз, факультет, направление, курс, форма обучения будет всплывающее окно с подсказкой, с возможностью написание текста для поиска.

4.2.5 Страница результата для зарегистрированного пользователя

Страница выдачи для вошедших пользователей будет содержать в себе кнопку «Главная» (в шапке), которая будет перенаправлять пользователя на главную страницу и кнопка Выход, перенаправляющая на главную страницу с завершением сессии. Так же присутствует меню фильтрации выдачи, спра-

ва от выдачи, там же будет кнопка “Очистить”, которая будет сбрасывать поля фильтрации и сами карточки с данными зарегистрированных студентов.

Фильтр для работодателя будет включать в себя: Курс, Форма обучения, Вуз, Факультет, Направление.

Фильтр для студента будет включать в себя: Курс, Форма обучения, Направление.

Для пунктов фильтрации будет всплывающее окно с подсказкой, с возможностью написание текста для поиска.

В карточках выдачи будут следующие данные: Фамилия, Имя, Отчество(если указано), Вуз, Факультет, Направление, Курс, Форма обучения, Контактные данные(по умолчанию email, остальные если есть).

Внешний вид сайта можно посмотреть на рисунках [7](#), [8](#), [9](#), [10](#), [11](#).



Рисунок 7 – Главная

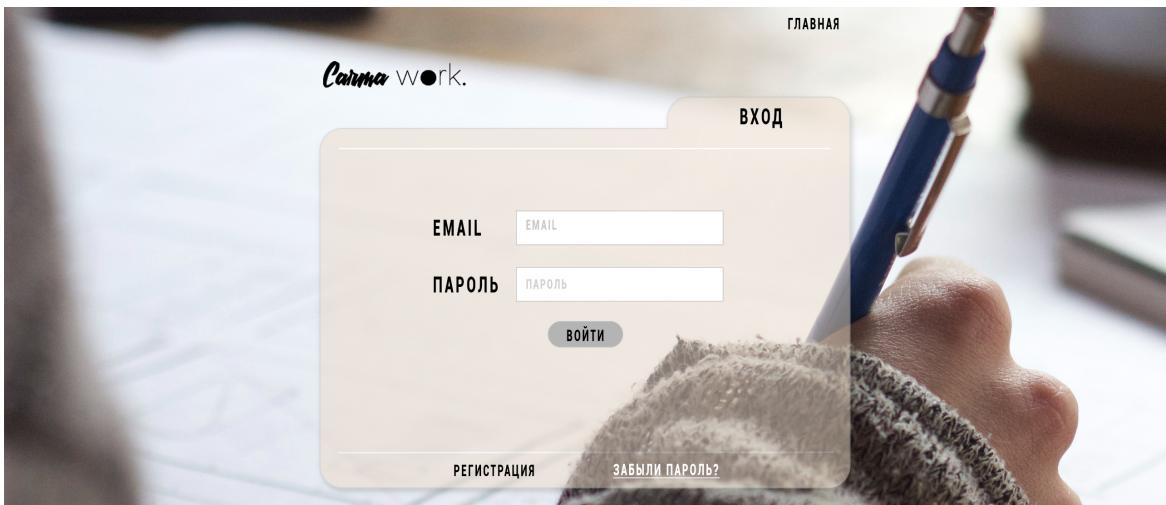


Рисунок 8 – Вход

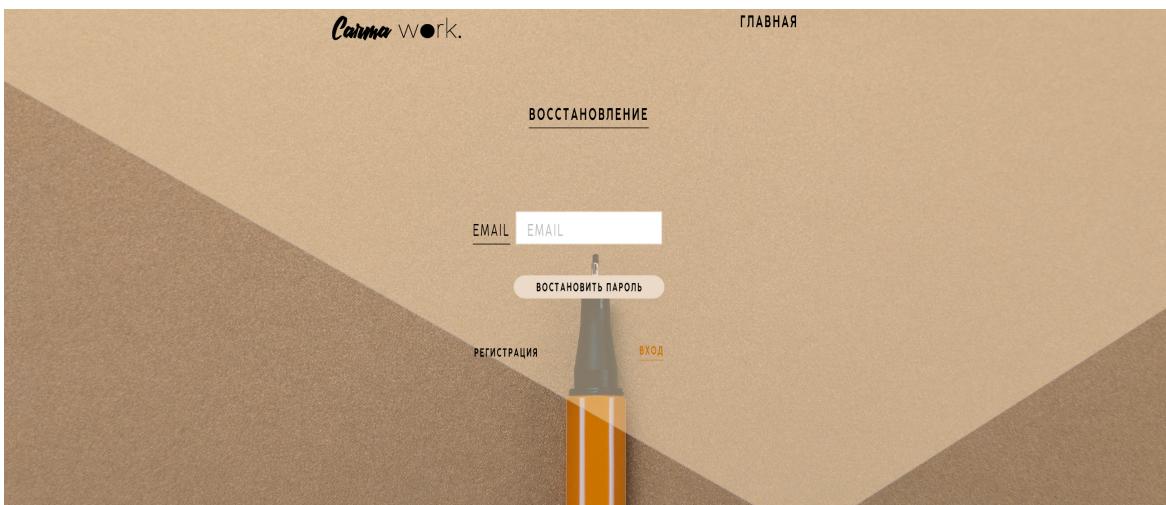


Рисунок 9 – Восстановление пароля



Рисунок 10 – Регистрация

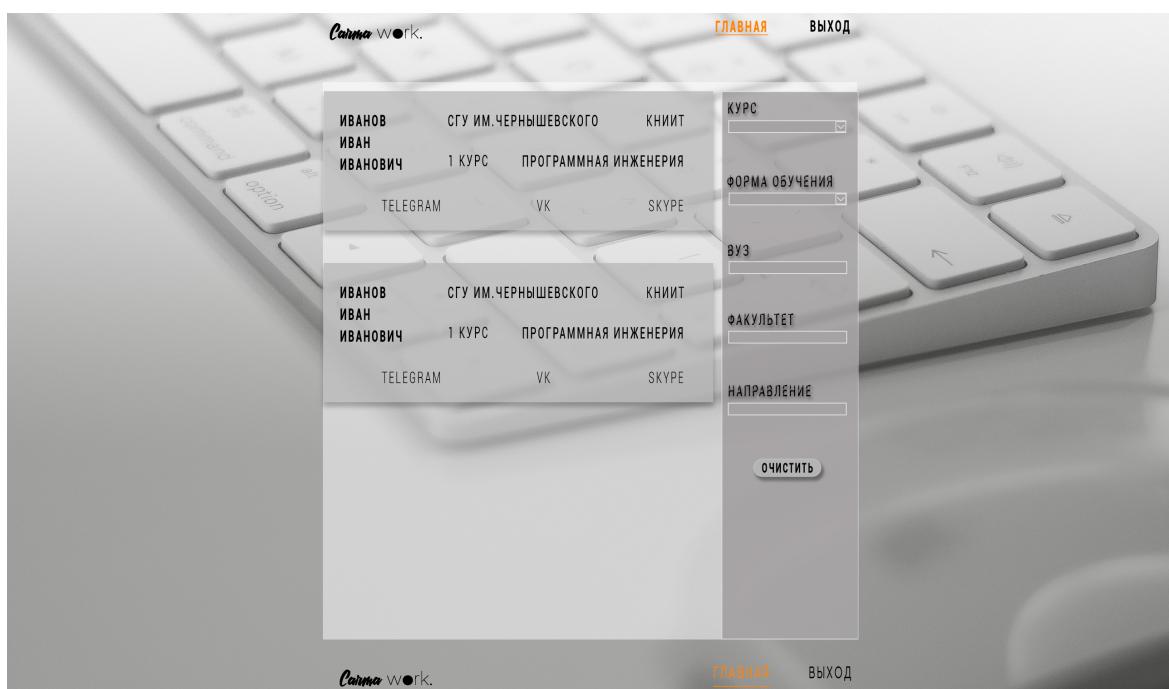


Рисунок 11 – Результат

5 Реализация базы данных и подключение к WEB интерфейсу

5.1 Реализация реляционной базы данных на языке - SQL

Для реализации базы данных на языке SQL, воспользуемся средой разработки MySQL.

Вначале был использован MySQL Workbench 8.0 для создания макета реляционной базы данных и приведения ее к 1 и 2 нормальному виду (См. рисунок 12).

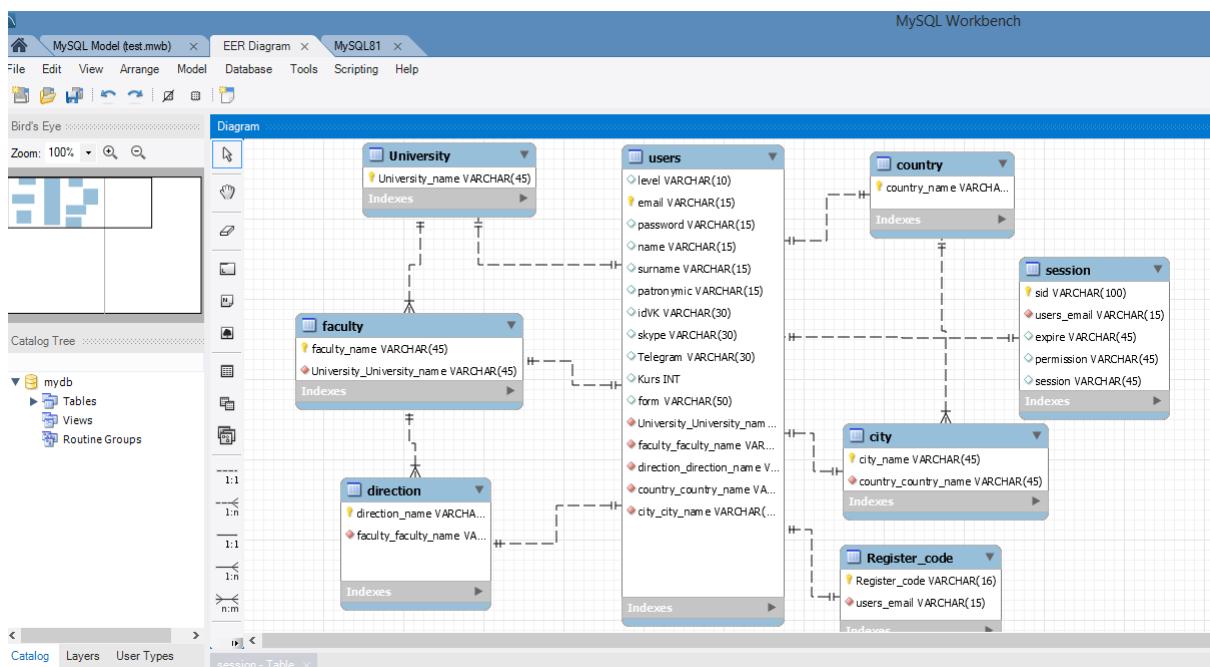
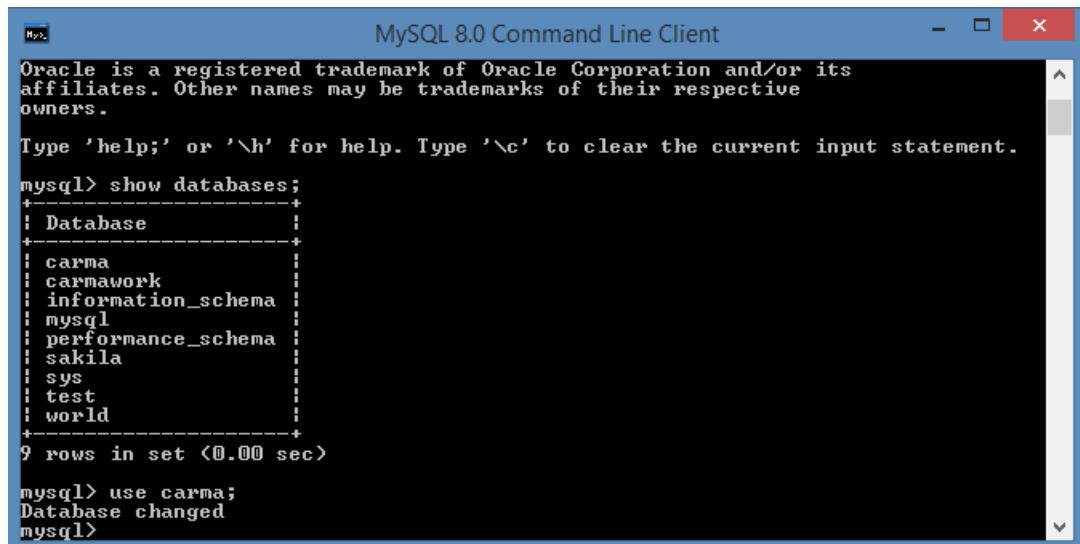


Рисунок 12 – MySql workbench 8.0

Далее воспользуемся MySql консолью для написания самой базы данных с помощью синтаксиса языка SQL. Была построена структура из 8 таблиц: university, faculty, direction, country, city, users, sessions, code. Статично были заполнены таблицы: university, faculty, direction, country, city; Остальные таблицы будут заполняться пользователем, по средством интерфейса, к которому далее будем подключать нашу базу данных, полный листинг см. приложение А). Структура получившейся базы данных см. рисунки 13, 14, 15, 16, 17, 18.



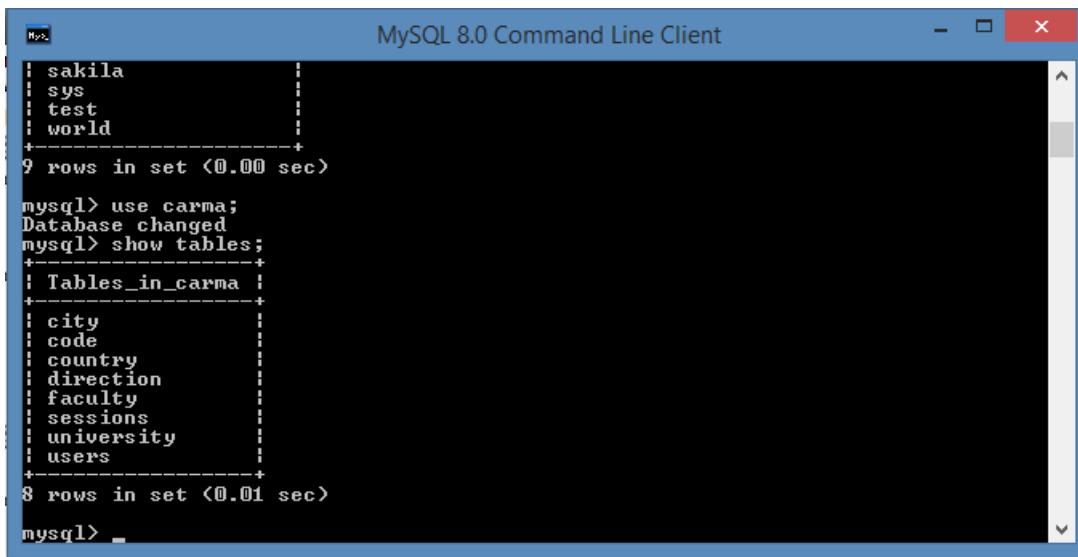
```
MySQL 8.0 Command Line Client
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| carma    |
| carmawork |
| information_schema |
| mysql     |
| performance_schema |
| sakila   |
| sys      |
| test     |
| world   |
+-----+
9 rows in set (0.00 sec)

mysql> use carma;
Database changed
mysql>
```

Рисунок 13 – База данных



```
MySQL 8.0 Command Line Client
sakila      |
sys        |
test       |
world      |
+-----+
9 rows in set (0.00 sec)

mysql> use carma;
Database changed
mysql> show tables;
+-----+
| Tables_in_carma |
+-----+
| city           |
| code           |
| country        |
| direction      |
| faculty        |
| sessions       |
| university     |
| users          |
+-----+
8 rows in set (0.01 sec)

mysql>
```

Рисунок 14 – Таблицы

```
0 rows in set (0.01 sec)

mysql> describe city;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| city_name | varchar(100) | NO | PRI | NULL |       |
| city_country | varchar(100) | YES | MUL | NULL |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.22 sec)

mysql> describe country;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| country_name | varchar(100) | NO | PRI | NULL |       |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

Рисунок 15 – Колонки

```
MySQL 8.0 Command Line Client

mysql> describe users;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| permission | varchar(100) | NO | PRI | NULL |       |
| email | varchar(100) | NO |       | NULL |       |
| password | varchar(100) | NO |       | NULL |       |
| name | varchar(100) | NO |       | NULL |       |
| surname | varchar(100) | NO |       | NULL |       |
| patronymic | varchar(100) | YES |       | NULL |       |
| user_country | varchar(100) | YES | MUL | NULL |       |
| user_city | varchar(100) | YES | MUL | NULL |       |
| id_vk | varchar(100) | YES |       | NULL |       |
| skype | varchar(100) | YES |       | NULL |       |
| telegram | varchar(100) | YES |       | NULL |       |
| kurs | varchar(100) | YES |       | NULL |       |
| form | varchar(100) | YES |       | NULL |       |
| user_university | varchar(100) | YES | MUL | NULL |       |
| user_faculty | varchar(100) | YES | MUL | NULL |       |
| user_direction | varchar(100) | YES | MUL | NULL |       |
+-----+-----+-----+-----+-----+-----+
16 rows in set (0.06 sec)

mysql>
```

Рисунок 16 – Колонки

```

mysql> describe university;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| university_name | varchar(100) | NO | PRI | NULL |       |
+-----+-----+-----+-----+-----+
1 row in set <0.01 sec>

mysql> describe faculty;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| faculty_name | varchar(100) | NO | PRI | NULL |       |
| faculty_university | varchar(100) | NO | MUL | NULL |       |
+-----+-----+-----+-----+-----+
2 rows in set <0.00 sec>

mysql> describe direction;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| direction_name | varchar(100) | NO | PRI | NULL |       |
| direction_faculty | varchar(100) | NO | MUL | NULL |       |
+-----+-----+-----+-----+-----+

```

Рисунок 17 – Колонки

```

mysql> describe code;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| code_name | varchar(100) | YES | NULL |
| email | varchar(100) | NO | PRI | NULL |       |
+-----+-----+-----+-----+-----+
2 rows in set <0.01 sec>

mysql> describe sessions;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| sessions_email | varchar(100) | YES | MUL | NULL |
| session | varchar(100) | YES | NULL |
| sid | varchar(100) | NO | PRI | NULL |
| expires | varchar(100) | YES | NULL |
| permission | varchar(100) | YES | NULL |
+-----+-----+-----+-----+-----+
5 rows in set <0.00 sec>

mysql>

```

Рисунок 18 – Колонки

5.2 Подключение базы данных к WEB интерфейсу

Для подключения базы данных к Web интерфейсу был использован сервер(API), написанный на node.js, в котором для работы базы данных были подключены следующие плагины:

- Express - framework, который использовали для работоспособной сессии; [7]
- Mysql - plugin, подключает базу данных к серверу; [8]
- connect-mysql - создает автоматизированное занесение в базу данных сессии пользователей перешедших на сайт. [9]

Далее по пунктам рассмотрим: подключение, создание сессии и запросы к базе(добавление, удаление, обновление).

5.2.1 Подключение

В начале инициализируем фреймворки и плагины:

```
1 var express = require('express');//веб-фреймворк
2 var bodyParser = require('body-parser');//обработка post запросов
3 var app = express();//создаем фреймворк
4 var mysql = require('mysql');//база данных
5 var domen = 'localhost';//домен по которому будет доступен сервер
6
7 var nodemailer = require('nodemailer');//обработчик почты
8
9 var session = require('express-session');//сессия
10 var MySQLStore = require('connect-mysql')(session);
```

следом подключаем нашу базу данных к серверу:

```
1 var options = {
2     config: {
3         user: 'root',
4         password: 'Facethree!23',
5         database: 'carma'
6     }
7 };
8
9 connection.connect(function(err, database){
10     if (err){
11         console.log('Не возможно подключиться к серверу MySQL. Ошибка: ', err);
12     }
13     else{
14         console.log('Соединение с базой данных успешно установлено');
15     }
16 });
17
18 })
```

5.2.2 Сессия

Конфигурация подключения сессии к MySql серверу и базе данных:

```
1 var options = {
2     config: {
3         user: 'root',
4         password: 'Facethree!23',
5         database: 'carma'
6     }
7 };
```

Настройка сессии:

```
1 app.use(session({//настройки сессии
2     secret: 'adth6e6',//ключ по которому будут шифроваться cookie
3     store: new MySQLStore(options)
4 }));
```

5.2.3 Запросы к базе данных

Запросы к базе данных из интерфейса и сервера были написаны на основе плагина mysql для node.js. Функция "connection.query" принимает в качестве аргумента синтаксис запросов SQL, пример:

```
1 connection.query('SELECT * FROM sessions WHERE sid = "' + req.sessionID +
  '", function(err, docs){
```

В данном примере выбрали из таблицы sessions колонку sid со значением переменной req.sessionId, значение которой генерируется при попадание пользователя на сайт, плагином express. Еще пример выбора данных:

```
1 connection.query('SELECT * FROM sessions WHERE sid = "' + req.sessionID +
  '", function(err, docs){
```

В этом примере мы выбираем из базы данных данные для страницы выдачи. Пример выбора данных для поиска в базе данных пользователей по email:

```
1 connection.query('SELECT * FROM users WHERE email = "' + user.email + '"',
  function(err, docs){// поиск в базе данных пользователей по email
```

далее будут продемонстрированы примеры добавления и обновления с более сложным синтаксисом:

```
1 connection.query("INSERT INTO users (permission, email, password, name,
  surname, patronymic, id_vk, skype, telegram, kurs, form) VALUES('" +
  newusers.permission + "', '" + newusers.email + "', '" +
  newusers.password + "', '" + newusers.name + "', '" +
  newusers.surname + "', '" + newusers.otche + "', '" + newusers.vk +
  "', '" + newusers.skype + "', '" + newusers.telegram + "', '" +
  newusers.course + "', '" + newusers.type + "')");
2 connection.query("UPDATE users SET user_university = (SELECT * FROM
  university WHERE university_name = '" + newusers.university + "')"
  WHERE email = '" + newusers.email + "'");
3 connection.query("UPDATE users SET user_faculty = (SELECT faculty_name
  FROM faculty WHERE faculty_name = '" + newusers.faculty + "')"
  WHERE email = '" + newusers.email + "'");
4 connection.query("UPDATE users SET user_direction = (SELECT
  direction_name FROM direction WHERE direction_name = '" +
  newusers.direction + "')"
  WHERE email = '" + newusers.email + "')";
```

В данном примере, в запросе INSERT, после обработки полей заполненных пользователем отправляем данные с форм в таблицу users в соответствие с заполненными полями, кроме вуза, факультета и направления, т.к у нас эти значение были статично внесены в базу данных и будут подгружаться из других таблиц по внешнему ключу, то делаем для каждого поля запрос на

UPDATE(обновление) данных в таблице user , колонке useruniversity, запрашивая статичные данные из таблицы university и его ключа universityname. И пример удаления:

```
1 app . post ( '/logout' , function ( req , res ) { //ответ на запрос разрыва авторизации
2     var status = {};
3     connection . query ( "DELETE FROM sessions WHERE sid = '" + req . sessionID + "' " );
4     res . send ( status );
5 } );
```

В данном примере с помощью DELETE удаляем из таблицы sessions строки, в которых значения колонки sid = значение сессии определенного пользователя, сделано это для разрыва авторизации, по нажатию кнопки "выход". Ввиду большого объема запросов, реализованных в приложении, в основной части были продемонстрированы только некоторые из них, полные тексты всех запросов находятся в приложении **Б**

ЗАКЛЮЧЕНИЕ

В данной курсовой работе была изучена теория реляционных баз данных, основы языка манипулирования данными - SQL. В качестве примера реализации реляционной базы данных была выбрана предметная область, связанная с поиском студентов: для работодателей и других студентов. Создано клиентское приложение в виде WEB интерфейса. Для этого клиентского приложения был создан ряд типовых запросов на языке SQL [10]. Приложение было реализовано с использованием:

- языка SQL в среде MySql для реализации реляционной базы данных;
- HTML5, CSS3(SASS), NODE.JS (Gulp + plugins), Bootstrap 4, JQuery для создания интерфейса базы данных;
- Node.js(express, mysql) для подключения СУБД к интерфейсу.

Во время проделанной работы, был сделан вывод, что не смотря на все свои преимущества реляционная модель базы данных, имеет и свои недостатки. Преимущества реляционной базы данных:

- гибкость – с таблицей можно производить любые операции склеивания и разрезания, т.о. можно получать информацию в любом виде. Один объект связывается с другим по столбцам и строкам, мы получаем 1 большую таблицу, из которой мы берем строки по нужным нам характеристикам. Эту таблицу мы можем склеивать и разделять.
- достаточно простой язык манипулирования данными. SQL – структурный язык запросов.
- точность – описание связей между таблицами и характеристиками. Можно описывать с помощью, так называемой алгебры отношений.

Недостатки реляционной базы данных:

- создание дополнительных таблиц, для внешних ключей. Это связано с тем, что поля одной таблицы должны содержать постоянное число полей
- высокая трудоемкость манипулирования информацией и изменения связей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 MySQL [Электронный ресурс]. — URL:<https://dev.mysql.com/doc/> (Дата обращения и время обращения 13.05.2018 16:00). Загл. с экрана. Яз. рус.
- 2 Райордан, Р. Основы реляционных баз данных / Р. Райордан. — 2001. — Р. 353.
- 3 Кузин, А. В. Базы данных / А. В. Кузин. — 2005. — Р. 320.
- 4 Архитектура СУБД [Электронный ресурс]. — URL:<http://zametkinapolyah.ru/zametki-o-mysql/arxitektura-subd-arxitektura-baz-dannix-logicheskaya-struktura-subd-opisanie-dannix-v-baze-dannix-bazy-dannix-sxema-dannix.html> (Дата обращения и время обращения 13.05.2018 18:00). Загл. с экрана. Яз. рус.
- 5 Томас Коннолли Каролин Бегг, А. С. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / А. С. Томас Коннолли, Каролин Бегг. — 2003. — Р. 1427.
- 6 Дейт,. ВВЕДЕНИЕ В системы баз данных / Дейт. — 2001. — Р. 1072.
- 7 Express plugin [Электронный ресурс]. — URL:<http://expressjs.com/> (Дата обращения и время обращения 13.05.2018 13:00). Загл. с экрана. Яз. рус.
- 8 MySql plugin[Электронный ресурс]. — URL:<https://www.npmjs.com/package/mysql> (Дата обращения и время обращения 13.05.2018 15:00). Загл. с экрана. Яз. рус.
- 9 Mysql Connector [Электронный ресурс]. — URL:<https://www.npmjs.com/package/connect-mysql> (Дата обращения и время обращения 13.05.2018 14:00). Загл. с экрана. Яз. рус.
- 10 SQL [Электронный ресурс]. — URL:<http://www.sql.ru/docs/sql/> (Дата обращения и время обращения 12.05.2018 18:00). Загл. с экрана. Яз. рус.

ПРИЛОЖЕНИЕ А

Список команд для реализации базы данных

Листинг SQL-listing.txt

```
1
2 create database carma;
3
4 create table university (
5 university_name varchar(100) NOT NULL,
6 PRIMARY KEY (university_name));
7
8 create table faculty (
9 faculty_name varchar(100) NOT NULL,
10 PRIMARY KEY (faculty_name),
11 faculty_university varchar(100) NOT NULL,
12 FOREIGN KEY (faculty_university) REFERENCES university (university_name)
13 );
14
15 create table direction (
16 direction_name varchar(100) NOT NULL,
17 PRIMARY KEY (direction_name),
18 direction_faculty varchar(100) NOT NULL,
19 FOREIGN KEY (direction_faculty) REFERENCES faculty (faculty_name)
20 );
21
22 create table country (
23 country_name varchar(100),
24 PRIMARY KEY (country_name));
25
26 create table city (
27 city_name varchar(100),
28 PRIMARY KEY (city_name),
29 city_country varchar(100),
30 FOREIGN KEY (city_country) REFERENCES country (country_name)
31 );
32
33 create table code (
34 code_name varchar(30),
35 PRIMARY KEY (code_name));
36
37 create table users (
38 level varchar(100) NOT NULL,
39 email varchar(100) NOT NULL,
40 PRIMARY KEY (email),
41 password varchar(100) NOT NULL,
42 name varchar(100) NOT NULL,
43 surname varchar(100) NOT NULL,
44 patronymic varchar(100),
```

```

45 user_country varchar(100) ,
46 FOREIGN KEY (user_country) REFERENCES country (country_name) ,
47 user_city varchar(100) ,
48 FOREIGN KEY (user_city) REFERENCES city (city_name) ,
49 id_vk varchar(100) ,
50 skype varchar(100) ,
51 telegram varchar(100) ,
52 kurs varchar(100) ,
53 form varchar(100) ,
54 user_university varchar(100) ,
55 FOREIGN KEY (user_university) REFERENCES university (university_name) ,
56 user_faculty varchar(100) ,
57 FOREIGN KEY (user_faculty) REFERENCES faculty (faculty_name) ,
58 user_direction varchar(100) ,
59 FOREIGN KEY (user_direction) REFERENCES direction (direction_name)
60 );
61
62 create table sessions (
63 sessions_id int ,
64 PRIMARY KEY (sessions_id) ,
65 sessions_email varchar(100) ,
66 FOREIGN KEY (sessions_email) REFERENCES users (email)
67 );
68
69
70
71 INSERT INTO university VALUES ( 'Саратовский Государственный Университет имени
    Н. Г. Чернышевского' );
72 INSERT INTO university VALUES ( 'Саратовский государственный технический
    университет имени Ю. А. Гагарина' );
73 INSERT INTO university VALUES ( 'Саратовская государственная юридическая
    академия' );
74
75
76
77 INSERT INTO faculty VALUES ( 'Биологический факультет' , 'Саратовский
    Государственный Университет имени Н. Г. Чернышевского' );
78 INSERT INTO faculty VALUES ( 'Географический факультет' , 'Саратовский
    Государственный Университет имени Н. Г. Чернышевского' );
79 INSERT INTO faculty VALUES ( 'Геологический факультет' , 'Саратовский
    Государственный Университет имени Н. Г. Чернышевского' );
80 INSERT INTO faculty VALUES ( 'Механико–математический факультет' , 'Саратовский
    Государственный Университет имени Н. Г. Чернышевского' );
81 INSERT INTO faculty VALUES ( 'Социологический факультет' , 'Саратовский
    Государственный Университет имени Н. Г. Чернышевского' );
82 INSERT INTO faculty VALUES ( 'Факультет компьютерных наук и информационных
    технологий' , 'Саратовский Государственный Университет имени Н. Г.
    Чернышевского' );

```

```
83 INSERT INTO faculty VALUES ('Факультет нано- и биомедицинских технологий',  
    'Саратовский Государственный Университет имени Н. Г. Чернышевского');  
84 INSERT INTO faculty VALUES ('Факультет нелинейных процессов', 'Саратовский  
    Государственный Университет имени Н. Г. Чернышевского');  
85 INSERT INTO faculty VALUES ('Факультет иностранных языков и лингводидактики',  
    'Саратовский Государственный Университет имени Н. Г. Чернышевского');  
86 INSERT INTO faculty VALUES ('Факультет психолого-педагогического и специального  
    образования', 'Саратовский Государственный Университет имени Н. Г.  
    Чернышевского');  
87 INSERT INTO faculty VALUES ('Факультет психологии', 'Саратовский  
    Государственный Университет имени Н. Г. Чернышевского');  
88 INSERT INTO faculty VALUES ('Физический факультет', 'Саратовский  
    Государственный Университет имени Н. Г. Чернышевского');  
89 INSERT INTO faculty VALUES ('Философский факультет', 'Саратовский  
    Государственный Университет имени Н. Г. Чернышевского');  
90 INSERT INTO faculty VALUES ('Экономический факультет', 'Саратовский  
    Государственный Университет имени Н. Г. Чернышевского');  
91 INSERT INTO faculty VALUES ('Юридический факультет', 'Саратовский  
    Государственный Университет имени Н. Г. Чернышевского');  
92 INSERT INTO faculty VALUES ('Институт искусств', 'Саратовский Государственный  
    Университет имени Н. Г. Чернышевского');  
93 INSERT INTO faculty VALUES ('Юридический факультет', 'Саратовский  
    Государственный Университет имени Н. Г. Чернышевского');  
94 INSERT INTO faculty VALUES ('Институт истории и международных отношений',  
    'Саратовский Государственный Университет имени Н. Г. Чернышевского');  
95 INSERT INTO faculty VALUES ('Институт физической культуры и спорта',  
    'Саратовский Государственный Университет имени Н. Г. Чернышевского');  
96 INSERT INTO faculty VALUES ('Институт филологии и журналистики', 'Саратовский  
    Государственный Университет имени Н. Г. Чернышевского');  
97 INSERT INTO faculty VALUES ('Институт химии', 'Саратовский Государственный  
    Университет имени Н. Г. Чернышевского');  
98 INSERT INTO faculty VALUES ('Институт дополнительного профессионального  
    образования', 'Саратовский Государственный Университет имени Н. Г.  
    Чернышевского');  
99  
100 INSERT INTO faculty VALUES ('Институт прикладных информационных технологий',  
    'Саратовский государственный технический университет имени Ю. А. Гагарина');  
101  
102  
103 INSERT INTO direction VALUES ('ИВЧТ', 'Институт прикладных информационных  
    технологий');  
104  
105 INSERT INTO direction VALUES ('Биология 06.03.01 (Бакалавриат)', 'Биологический  
    факультет');  
106 INSERT INTO direction VALUES ('Биология 06.04.01 (Магистратура)',  
    'Биологический факультет');  
107 INSERT INTO direction VALUES ('Биология 06.06.01 (Аспирантура)', 'Биологический  
    факультет');
```

```
108 INSERT INTO direction VALUES ('Биология 06.05.01 (Специалитет)', 'Биологический  
факультет');  
109 INSERT INTO direction VALUES ('Педагогическое образование 44.03.01  
(Бакалавриат)', 'Биологический факультет');  
110 INSERT INTO direction VALUES ('Педагогическое образование 44.04.01  
(Магистратура)', 'Биологический факультет');  
111  
112 INSERT INTO direction VALUES ('География 05.03.02 (Бакалавриат)',  
'Географический факультет');  
113 INSERT INTO direction VALUES ('География 05.04.02 (Магистратура)',  
'Географический факультет');  
114 INSERT INTO direction VALUES ('Экология и природопользование 05.03.06  
(Бакалавриат)', 'Географический факультет');  
115 INSERT INTO direction VALUES ('Экология и природопользование 05.04.06  
(Магистратура)', 'Географический факультет');  
116 INSERT INTO direction VALUES ('Науки о земле 05.06.01 (Аспирантура)',  
'Географический факультет');  
117 INSERT INTO direction VALUES ('Прикладная гидрометеорология 05.03.05  
(Бакалавриат)', 'Географический факультет');  
118 INSERT INTO direction VALUES ('Прикладная гидрометеорология 05.04.05  
(Магистратура)', 'Географический факультет');  
119 INSERT INTO direction VALUES ('Педагогическое образование 44.03.01  
(Бакалавриат)', 'Географический факультет');  
120 INSERT INTO direction VALUES ('Прикладная информатика 09.03.03 (Бакалавриат)',  
'Географический факультет');  
121 INSERT INTO direction VALUES ('Картография и геоинформатика 05.03.03  
(Бакалавриат)', 'Географический факультет');  
122  
123 INSERT INTO direction VALUES ('Геология 05.03.01 (Бакалавриат)', 'Геологический  
факультет');  
124 INSERT INTO direction VALUES ('Геология 05.04.01 (Магистратура)',  
'Геологический факультет');  
125 INSERT INTO direction VALUES ('Геология 05.06.01 (Аспирантура)', 'Геологический  
факультет');  
126 INSERT INTO direction VALUES ('Нефтегазовое дело 21.03.01 (Бакалавриат)',  
'Геологический факультет');  
127 INSERT INTO direction VALUES ('Прикладная геология 21.05.02 (Специалитет)',  
'Геологический факультет');  
128  
129 INSERT INTO direction VALUES ('Прикладная математика и информатика 01.03.02  
(Бакалавриат)', 'Механико-математический факультет');  
130 INSERT INTO direction VALUES ('Социология 39.03.01 (Бакалавриат)',  
'Социологический факультет');  
131 INSERT INTO direction VALUES ('Электроника и наноэлектроника 11.03.04  
(Бакалавриат)', 'Факультет нано- и биомедицинских технологий');  
132  
133 INSERT INTO direction VALUES ('Информатика и вычислительная техника 09.03.01  
(Бакалавриат)', 'Факультет компьютерных наук и информационных технологий');
```

```
134 INSERT INTO direction VALUES ('Информатика и вычислительная техника 09.04.01  
        (Магистратура)', 'Факультет компьютерных наук и информационных технологий');  
135 INSERT INTO direction VALUES ('Информатика и вычислительная техника 09.06.01  
        (Аспирантура)', 'Факультет компьютерных наук и информационных технологий');  
136 INSERT INTO direction VALUES ('Программная инженерия 09.03.04 (Бакалавриат)',  
        'Факультет компьютерных наук и информационных технологий');  
137 INSERT INTO direction VALUES ('Системный анализ и управление 27.03.03  
        (Бакалавриат)', 'Факультет компьютерных наук и информационных технологий');  
138 INSERT INTO direction VALUES ('Математика и механика 01.06.01 (Аспирантура)',  
        'Факультет компьютерных наук и информационных технологий');  
139 INSERT INTO direction VALUES ('Математическое обеспечение и администрирование  
        информационных систем 02.03.03 (Бакалавриат)', 'Факультет компьютерных наук  
        и информационных технологий');  
140 INSERT INTO direction VALUES ('Фундаментальная информатика и информационные  
        технологии 02.03.02 (Бакалавриат)', 'Факультет компьютерных наук и  
        информационных технологий');  
141 INSERT INTO direction VALUES ('Прикладная математика и информатика 01.04.02  
        (Магистратура)', 'Факультет компьютерных наук и информационных технологий');  
142 INSERT INTO direction VALUES ('Компьютерная безопасность (Специалитет) 10.05.01  
        (Специалитет)', 'Факультет компьютерных наук и информационных технологий');  
143 INSERT INTO direction VALUES ('Компьютерные и информационные науки 02.06.01  
        (Аспирантура)', 'Факультет компьютерных наук и информационных технологий');  
144  
145 INSERT INTO direction VALUES ('Прикладные математика и физика 03.03.01  
        (Бакалавриат)', 'Факультет нелинейных процессов');  
146 INSERT INTO direction VALUES ('Психолого–педагогическое образование 44.03.02  
        (Бакалавриат)', 'Факультет психолого–педагогического и специального  
        образования');  
147 INSERT INTO direction VALUES ('Психология 37.03.01 (Бакалавриат)', 'Факультет  
        психологии');  
148 INSERT INTO direction VALUES ('Физика 03.03.02 (Бакалавриат)', 'Физический  
        факультет');  
149 INSERT INTO direction VALUES ('Культурология 51.03.01 (Бакалавриат)',  
        'Философский факультет');  
150 INSERT INTO direction VALUES ('Экономика 38.03.01 (Бакалавриат)',  
        'Экономический факультет');  
151 INSERT INTO direction VALUES ('Юриспруденция 40.03.01 (Бакалавриат)',  
        'Юридический факультет');  
152 INSERT INTO direction VALUES ('Народная художественная культура 51.03.02  
        (Бакалавриат)', 'Институт искусств');  
153  
154  
155 INSERT INTO country VALUES ('Российская Федерация');  
156 INSERT INTO country VALUES ('Украина');  
157 INSERT INTO country VALUES ('Польша');  
158  
159  
160
```

```
161 INSERT INTO city VALUES ( 'Варшава' , 'Польша' );
162 INSERT INTO city VALUES ( 'Саратов' , 'Российская Федерация' );
163 INSERT INTO city VALUES ( 'Самара' , 'Российская Федерация' );
164 INSERT INTO city VALUES ( 'Волгоград' , 'Российская Федерация' );
165 INSERT INTO city VALUES ( 'Киев' , 'Украина' );
```

ПРИЛОЖЕНИЕ Б

Программный код для подключения базы данных к WEB интерфейсу

Листинг app.js

```
1 var express = require('express');//веб-фреймворк
2 var bodyParser = require('body-parser');//обработка post запросов
3 var app = express();//создаем фреймворк
4 var mysql = require('mysql');//база данных
5 var domen = 'localhost';//домен по которому будет доступен сервер
6
7 var nodemailer = require('nodemailer');//обработчик почты
8
9 var session = require('express-session');//сессия
10 var MySQLStore = require('connect-mysql')(session);
11
12 var connection = mysql.createConnection({
13     host      : 'localhost',
14     user      : 'root',
15     password  : 'Facethree!23',
16     database  : 'carma'
17 });
18
19 var options = {
20     config: {
21         user: 'root',
22         password: 'Facethree!23',
23         database: 'carma'
24     }
25 };
26
27 app.use(session({//настройки сессии
28     secret: 'adth6e6',//ключ по которому будут шифроваться cookie
29     store: new MySQLStore(options)
30 }));
31
32 app.use(bodyParser.json());//подключаем к фреймворку обработку post запросов
33 app.use(bodyParser.urlencoded({ extended: true}));//подключаем обработку данных
            из пост запросов
34 app.use('/', express.static('dist'));//разрешаем нашим страницам загружать
            дополнительные файлы (css, js, img)
35
36 app.listen(8081, domen, function () {//запуск сервера по 80 порту и указанному
            домену
37     console.log('Успешно запущен');
38 });
39
40 connection.connect(function(err, database){
```

```

41  if (err) {
42      console.log('Не возможно подключиться к серверу MySQL. Ошибка:', err);
43  }
44  else{
45      console.log('Соединение с базой данных успешно установлено');
46  }
47 });
48
49 app.get('/', function (req, res) { //ответ на запрос главной страницы
50     res.sendFile(__dirname + '/dist/index.html'); //отправляем нашу главную
51     //страницу
52     console.log('Переход на главную');
53 });
54
55 app.get('/login', function (req, res) { //ответ на запрос страницы входа
56     connection.query('SELECT * FROM sessions WHERE sid = "' + req.sessionID +
57     '"', function(err, docs){
58         if (err){
59             console.log(err);
60         }
61         else if (docs[0]){
62             if (docs[0].sessions_email == null){
63                 res.sendFile(__dirname + '/dist/login.html'); //отправляем страницу
64                 //входа
65                 console.log('Переход на вход');
66             }
67             else{
68                 res.redirect('/view'); //перенаправление на выдачу
69                 console.log('Переход на выдачу');
70             }
71         }
72     });
73
74     app.get('/register', function (req, res) { //ответ на запрос страницы регистрации
75         connection.query('SELECT * FROM sessions WHERE sid = "' + req.sessionID +
76         '"', function(err, docs){
77             if (err){
78                 console.log(err);
79             }
80             else if (docs[0]){
81                 if (docs[0].sessions_email == null){
82                     res.sendFile(__dirname + '/dist/register.html'); //отправляем
83                     //страницу регистрации
84                     console.log('Переход на регистрацию');
85                 }
86                 else{
87                     res.redirect('/view'); //перенаправление на выдачу

```

```

84         console.log('Переход на выдачу');
85     }
86   }
87 );
88 });
89
90 app.get('/view', function (req, res) { //ответ на запрос страницы выдачи
91   connection.query('SELECT * FROM sessions WHERE sid = "' + req.sessionID +
92     '"', function(err, docs){
93     if (err){
94       console.log(err);
95     } else if (docs[0]){
96       if (docs[0].sessions_email == null){
97         res.redirect('/login');//перенаправление на вход
98         console.log('Переход на вход');
99       }
100      else{
101        res.sendFile(__dirname + '/dist/result.html');//отправляем страницу
102          выдачи
103          console.log('Переход на выдачу');
104      }
105    });
106  });
107
108 app.get('/rebootpass', function (req, res) { //ответ на запрос страницы
109   восстановления пароля
110   connection.query('SELECT * FROM sessions WHERE sid = "' + req.sessionID +
111     '"', function(err, docs){
112     if (err){
113       console.log(err);
114     } else if (docs[0]){
115       if (docs[0].sessions_email == null){
116         res.sendFile(__dirname + '/dist/rebootpass.html');//отправляем
117           страницу восстановления пароля
118           console.log('Переход на восстановление');
119       }
120     else{
121       res.sendFile(__dirname + '/dist/result.html');//отправляем страницу
122         выдачи
123         console.log('Переход на выдачу');
124     }
125   });
126 });

```

```

126 app.post('/login', function(req, res){//ответ на запрос страницы со страницы
    //хода
127     var user = { //полученные данные
128         email: req.body.login,
129         password: req.body.password
130     };
131     var status = { //статус проверки
132         login: '',
133         password: ''
134     };
135     connection.query('SELECT * FROM users WHERE email = "' + user.email + '"',
136         function(err, docs){ // поиск в базе данных пользователей по email
137             if(err){ //выход если ошибка
138                 console.log(err);
139             } else if(docs[0]){ //если найдено совпадение
140                 if(docs[0].email == user.email && docs[0].password ==
141                     user.password){ //если совпадает и логин и пароль
142                     connection.query("UPDATE sessions SET sessions_email = '" +
143                         user.email + "', WHERE sid = '" + req.sessionID + "'");
144                     connection.query("UPDATE sessions SET permission = '" +
145                         docs[0].permission + "' WHERE sid = '" + req.sessionID + "'");
146                     console.log('Успешный вход ' + user.email + ', ' +
147                         user.password); //выход в консоль аутентификацию пользователя
148                     status.login = true; //устанавливаем положительный статус
149                     status.password = true;
150                 } else{ //иначе
151                     status.login = true; //устанавливаем положительный статус логина
152                     status.password = false; //и отрицательный для пароля
153                     console.log('Не правильный пароль ' + user.email + ', ' +
154                         user.password); //выход в консоль не успешную аутентификацию
155                     //пользователя
156                 }
157             } else{ //иначе
158                 console.log('Пользователь ' + user.email + ', ' + user.password + ' не
159                 найден'); //выводим попытку аутентификации
160                 status.login = false; //устанавливаем отрицательный статус
161                 status.password = false;
162             }
163         }
164     });
165     res.send(status); //отправляем статус
166 });
167 });
168 });
169 });
170 });
171 });
172 });
173 });
174 });
175 });
176 });
177 });
178 });
179 });
180 });
181 });
182 });
183 });
184 });
185 });
186 });
187 });
188 });
189 });
190 });
191 });
192 });
193 });
194 });
195 });
196 });
197 });
198 });
199 });
200 });
201 });
202 });
203 });
204 });
205 });
206 });
207 });
208 });
209 });
210 });
211 });
212 });
213 });
214 });
215 });
216 });
217 });
218 });
219 });
220 });
221 });
222 });
223 });
224 });
225 });
226 });
227 });
228 });
229 });
230 });
231 });
232 });
233 });
234 });
235 });
236 });
237 });
238 });
239 });
240 });
241 });
242 });
243 });
244 });
245 });
246 });
247 });
248 });
249 });
250 });
251 });
252 });
253 });
254 });
255 });
256 });
257 });
258 });
259 });
260 });
261 });
262 });
263 });
264 });
265 });
266 });
267 });
268 });
269 });
270 });
271 });
272 });
273 });
274 });
275 });
276 });
277 });
278 });
279 });
280 });
281 });
282 });
283 });
284 });
285 });
286 });
287 });
288 });
289 });
290 });
291 });
292 });
293 });
294 });
295 });
296 });
297 });
298 });
299 });
299 });
300 });
301 });
302 });
303 });
304 });
305 });
306 });
307 });
308 });
309 });
309 });
310 });
311 });
312 });
313 });
314 });
315 });
316 });
317 });
318 });
319 });
319 });
320 });
321 });
322 });
323 });
324 });
325 });
326 });
327 });
328 });
329 });
329 });
330 });
331 });
332 });
333 });
334 });
335 });
336 });
337 });
338 });
339 });
339 });
340 });
341 });
342 });
343 });
344 });
345 });
346 });
347 });
348 });
349 });
349 });
350 });
351 });
352 });
353 });
354 });
355 });
356 });
357 });
358 });
359 });
359 });
360 });
361 });
362 });
363 });
364 });
365 });
366 });
367 });
368 });
369 });
369 });
370 });
371 });
372 });
373 });
374 });
375 });
376 });
377 });
378 });
379 });
379 });
380 });
381 });
382 });
383 });
384 });
385 });
386 });
387 });
388 });
389 });
389 });
390 });
391 });
392 });
393 });
394 });
395 });
396 });
397 });
398 });
399 });
399 });
400 });
401 });
402 });
403 });
404 });
405 });
406 });
407 });
408 });
409 });
409 });
410 });
411 });
412 });
413 });
414 });
415 });
416 });
417 });
418 });
419 });
419 });
420 });
421 });
422 });
423 });
424 });
425 });
426 });
427 });
428 });
429 });
429 });
430 });
431 });
432 });
433 });
434 });
435 });
436 });
437 });
438 });
439 });
439 });
440 });
441 });
442 });
443 });
444 });
445 });
446 });
447 });
448 });
449 });
449 });
450 });
451 });
452 });
453 });
454 });
455 });
456 });
457 });
458 });
459 });
459 });
460 });
461 });
462 });
463 });
464 });
465 });
466 });
467 });
468 });
469 });
469 });
470 });
471 });
472 });
473 });
474 });
475 });
476 });
477 });
478 });
479 });
479 });
480 });
481 });
482 });
483 });
484 });
485 });
486 });
487 });
488 });
489 });
489 });
490 });
491 });
492 });
493 });
494 });
495 });
496 });
497 });
498 });
499 });
499 });
500 });
501 });
502 });
503 });
504 });
505 });
506 });
507 });
508 });
509 });
509 });
510 });
511 });
512 });
513 });
514 });
515 });
516 });
517 });
518 });
519 });
519 });
520 });
521 });
522 });
523 });
524 });
525 });
526 });
527 });
528 });
529 });
529 });
530 });
531 });
532 });
533 });
534 });
535 });
536 });
537 });
538 });
539 });
539 });
540 });
541 });
542 });
543 });
544 });
545 });
546 });
547 });
548 });
549 });
549 });
550 });
551 });
552 });
553 });
554 });
555 });
556 });
557 });
558 });
559 });
559 });
560 });
561 });
562 });
563 });
564 });
565 });
566 });
567 });
568 });
569 });
569 });
570 });
571 });
572 });
573 });
574 });
575 });
576 });
577 });
578 });
579 });
579 });
580 });
581 });
582 });
583 });
584 });
585 });
586 });
587 });
588 });
589 });
589 });
590 });
591 });
592 });
593 });
594 });
595 });
596 });
597 });
598 });
599 });
599 });
600 });
601 });
602 });
603 });
604 });
605 });
606 });
607 });
608 });
609 });
609 });
610 });
611 });
612 });
613 });
614 });
615 });
616 });
617 });
618 });
619 });
619 });
620 });
621 });
622 });
623 });
624 });
625 });
626 });
627 });
628 });
629 });
629 });
630 });
631 });
632 });
633 });
634 });
635 });
636 });
637 });
638 });
639 });
639 });
640 });
641 });
642 });
643 });
644 });
645 });
646 });
647 });
648 });
649 });
649 });
650 });
651 });
652 });
653 });
654 });
655 });
656 });
657 });
658 });
659 });
659 });
660 });
661 });
662 });
663 });
664 });
665 });
666 });
667 });
668 });
669 });
669 });
670 });
671 });
672 });
673 });
674 });
675 });
676 });
677 });
678 });
679 });
679 });
680 });
681 });
682 });
683 });
684 });
685 });
686 });
687 });
688 });
689 });
689 });
690 });
691 });
692 });
693 });
694 });
695 });
696 });
697 });
698 });
699 });
699 });
700 });
701 });
702 });
703 });
704 });
705 });
706 });
707 });
708 });
709 });
709 });
710 });
711 });
712 });
713 });
714 });
715 });
716 });
717 });
718 });
719 });
719 });
720 });
721 });
722 });
723 });
724 });
725 });
726 });
727 });
728 });
729 });
729 });
730 });
731 });
732 });
733 });
734 });
735 });
736 });
737 });
738 });
739 });
739 });
740 });
741 });
742 });
743 });
744 });
745 });
746 });
747 });
748 });
749 });
749 });
750 });
751 });
752 });
753 });
754 });
755 });
756 });
757 });
758 });
759 });
759 });
760 });
761 });
762 });
763 });
764 });
765 });
766 });
767 });
768 });
769 });
769 });
770 });
771 });
772 });
773 });
774 });
775 });
776 });
777 });
778 });
779 });
779 });
780 });
781 });
782 });
783 });
784 });
785 });
786 });
787 });
788 });
789 });
789 });
790 });
791 });
792 });
793 });
794 });
795 });
796 });
797 });
798 });
799 });
799 });
800 });
801 });
802 });
803 });
804 });
805 });
806 });
807 });
808 });
809 });
809 });
810 });
811 });
812 });
813 });
814 });
815 });
816 });
817 });
818 });
819 });
819 });
820 });
821 });
822 });
823 });
824 });
825 });
826 });
827 });
828 });
829 });
829 });
830 });
831 });
832 });
833 });
834 });
835 });
836 });
837 });
838 });
839 });
839 });
840 });
841 });
842 });
843 });
844 });
845 });
846 });
847 });
848 });
849 });
849 });
850 });
851 });
852 });
853 });
854 });
855 });
856 });
857 });
858 });
859 });
859 });
860 });
861 });
862 });
863 });
864 });
865 });
866 });
867 });
868 });
869 });
869 });
870 });
871 });
872 });
873 });
874 });
875 });
876 });
877 });
878 });
879 });
879 });
880 });
881 });
882 });
883 });
884 });
885 });
886 });
887 });
888 });
889 });
889 });
890 });
891 });
892 });
893 });
894 });
895 });
896 });
897 });
898 });
899 });
899 });
900 });
901 });
902 });
903 });
904 });
905 });
906 });
907 });
908 });
909 });
909 });
910 });
911 });
912 });
913 });
914 });
915 });
916 });
917 });
918 });
919 });
919 });
920 });
921 });
922 });
923 });
924 });
925 });
926 });
927 });
928 });
929 });
929 });
930 });
931 });
932 });
933 });
934 });
935 });
936 });
937 });
938 });
939 });
939 });
940 });
941 });
942 });
943 });
944 });
945 });
946 });
947 });
948 });
949 });
949 });
950 });
951 });
952 });
953 });
954 });
955 });
956 });
957 });
958 });
959 });
959 });
960 });
961 });
962 });
963 });
964 });
965 });
966 });
967 });
968 });
969 });
969 });
970 });
971 });
972 });
973 });
974 });
975 });
976 });
977 });
978 });
979 });
979 });
980 });
981 });
982 });
983 });
984 });
985 });
986 });
987 });
988 });
989 });
989 });
990 });
991 });
992 });
993 });
994 });
995 });
996 });
997 });
998 });
999 });
999 });

```

```

163     var status = {};
164     connection.query("DELETE FROM sessions WHERE sid = '" + req.sessionID + "'");
165     res.send(status);
166 });
167
168 app.post('/button', function(req, res){//ответ на запрос информации о сессии
169     var status = {
170         session: '',
171     };
172     connection.query('SELECT * FROM sessions WHERE sid = "' + req.sessionID +
173         '"', function(err, docs){
174         if(err){
175             console.log(err);
176         } else if(docs[0]){
177             if(docs[0].sessions_email == null){
178                 status.session = false;
179             }
180             else{
181                 status.session = true;
182             }
183         }
184         res.send(status); //отправляем статус
185     });
186 });
187
188 app.post('/country', function(req, res){//ответ на запрос списка стран
189     connection.query('SELECT * FROM country', function(err, docs){
190         if(err){//выход если ошибка
191             console.log(err);
192         } else if(docs !== null){//если найдено соединение
193             var list = [];
194             docs.forEach(function(element){
195                 list.push(element.country_name);
196             });
197             res.send(list); //отправляем список стран
198         } else{//иначе выводим в консоль не найдено
199             console.log('не найдено');
200         }
201     });
202 });
203 });
204 });
205
206 app.post('/countrycheck', function(req, res){//ответ на запрос списка стран
207     var status = {
208         found: '',
209     };

```

```

210 connection.query('SELECT * FROM country WHERE country_name = "' +
211     req.body.val + '"', function(err, docs){// поиск в базе данных
212     // пользователей по email
213     if(err){// вывод если ошибка
214         console.log(err);
215     }
216     else if(docs[0]){// если найдено совпадение
217         if(docs[0].country_name == req.body.val){// если совпадает и логин и
218             // пароль
219             status.found = true; // присваиваем положительный статус
220         }
221     }
222     else{//иначе
223         status.found = false; // отрицательный статус
224         console.log('не найдено');// вывод в консоль не найдено
225     }
226     res.send(status); // отправка статуса
227 });
228
229 app.post('/city', function(req, res){// ответ на запрос списка городов
230     connection.query('SELECT * FROM city WHERE city_country = "' +
231         req.body.val +
232         '"', function(err, docs){
233         if(err){// вывод если ошибка
234             console.log(err);
235         }
236         else if(docs !== null){// если найдено совпадение
237             var list = [];
238             docs.forEach(function(element){
239                 list.push(element.city_name);
240             });
241             res.send(list); // отправляем список городов
242         }
243     });
244
245 app.post('/citycheck', function(req, res){// ответ на запрос списка стран
246     var status = {
247         found: ''
248     };
249     connection.query('SELECT * FROM city WHERE city_name = "' +
250         req.body.val +
251         '"', function(err, docs){// поиск в базе данных пользователей по email
252         if(err){// вывод если ошибка
253             console.log(err);
254         }

```

```

253     else if (docs[0]){//если найдено совпадение
254         if (docs[0].city_name == req.body.val){//если совпадает и логин и пароль
255             status.found = true;//присваиваем положительный статус
256         }
257     }
258     else{//иначе
259         status.found = false;//отрицательный статус
260         console.log('не найдено');//вывод в консоль не найдено
261     }
262     res.send(status);//отправка статуса
263 });
264 });
265
266 app.post('/university', function(req, res){//ответ на запрос списка
267     //университетов
268     connection.query('SELECT * FROM university', function(err, docs){
269         if (err){//выход если ошибка
270             console.log(err);
271         }
272         else if (docs !== null){//если найдено совпадение
273             var list = [];
274             docs.forEach(function(element){
275                 list.push(element.university_name);
276             });
277             res.send(list);//отправляем список университетов
278         }
279         else{//иначе выводим в консоль не найдено
280             console.log('не найдено');
281         }
282     });
283 });
284
285 app.post('/universitycheck', function(req, res){//ответ на запрос проверки
286     //университета
287     var status = {
288         found: ''
289     };
290     connection.query('SELECT * FROM university WHERE university_name = "' +
291         req.body.val + '"', function(err, docs){// поиск в базе данных
292         if (err){//выход если ошибка
293             console.log(err);
294         }
295         else if (docs[0]){//если найдено совпадение
296             if (docs[0].university_name == req.body.val){//если совпадает и логин и
297                 //пароль
298                 status.found = true;//присваиваем положительный статус
299             }
300         }
301     });
302 });

```

```

296     }
297     else{//иначе
298         status.found = false; //отрицательный статус
299         console.log('не найдено');//вывод в консоль не найдено
300     }
301     res.send(status); //отправка статуса
302 });
303 });
304
305 app.post('/faculty', function(req, res){//ответ на запрос списка факультетов
306     connection.query('SELECT * FROM faculty WHERE faculty_university = "' +
307         req.body.val + '"', function(err, docs){
308         if(err){//выход если ошибка
309             console.log(err);
310         }
311         else if(docs !== null){//если найдено совпадение
312             var list = [];
313             docs.forEach(function(element){
314                 list.push(element.faculty_name);
315             });
316             res.send(list); //отправляем список университетов
317         }
318         else{//иначе выводим в консоль не найдено
319             console.log('не найдено');
320         }
321     });
322 }
323 app.post('/facultycheck', function(req, res){//ответ на запрос проверки
324     var status = {
325         found: ''
326     };
327     connection.query('SELECT * FROM faculty WHERE faculty_name = "' +
328         req.body.val + '"', function(err, docs){//поиск в базе данных
329         if(err){//выход если ошибка
330             console.log(err);
331         }
332         else if(docs[0]){ //если найдено совпадение
333             if(docs[0].faculty_name === req.body.val){ //если совпадает и логин и
334                 status.found = true; //присваиваем положительный статус
335             }
336         }
337         else{//иначе
338             status.found = false; //отрицательный статус
339             console.log('не найдено');//выход в консоль не найдено

```

```

339         }
340         res.send(status); //отправка статуса
341     });
342 });
343
344 app.post('/direction', function(req, res){ //ответ на запрос списка направлений
345     connection.query('SELECT * FROM direction WHERE direction_faculty = "' +
346         req.body.val + '"', function(err, docs){
347         if(err){ //вывод если ошибка
348             console.log(err);
349         } else if(docs !== null){ //если найдено совпадение
350             var list = [];
351             docs.forEach(function(element){
352                 list.push(element.direction_name);
353             });
354             res.send(list); //отправляем список университетов
355         } else{ //иначе выводим в консоль не найдено
356             console.log('не найдено');
357         }
358     });
359 });
360 });
361
362 app.post('/directioncheck', function(req, res){ //ответ на запрос проверки
363     var status = {
364         found: ''
365     };
366     connection.query('SELECT * FROM direction WHERE direction_name = "' +
367         req.body.val + '"', function(err, docs){ //нousк в базе данных
368         if(err){ //вывод если ошибка
369             console.log(err);
370         } else if(docs[0]){ //если найдено совпадение
371             if(docs[0].direction_name == req.body.val){ //если совпадает и логин и
372                 пароль
373                 status.found = true; //присваиваем положительный статус
374             }
375         } else{ //иначе
376             status.found = false; //отрицательный статус
377             console.log('не найдено'); //вывод в консоль не найдено
378         }
379         res.send(status); //отправка статуса
380     });
381 });

```

```

382
383 app . post ( '/register' , function ( req , res ) { //ответ на запрос регистрации
384     var newusers ; //объект новый пользователь
385     if ( req . body . permission == 1 ) { //если студент
386         newusers = { //определяем поля студента
387             permission : req . body . permission ,
388             email : req . body . email ,
389             password : req . body . password ,
390             name : req . body . name ,
391             surname : req . body . surname ,
392             otche : req . body . otche ,
393             country : req . body . country , //связанное поле
394             city : req . body . city , //связанное поле
395             vk : req . body . vk ,
396             skype : req . body . skype ,
397             telegram : req . body . telegram ,
398             course : req . body . course ,
399             type : req . body . type ,
400             university : req . body . university , //связанное поле
401             faculty : req . body . faculty , //связанное поле
402             direction : req . body . direction //связанное поле
403         } ;
404         connection . query ( "INSERT INTO users ( permission , email , password , name ,
405             surname , patronymic , id_vk , skype , telegram , kurs , form ) VALUES ( " +
406             newusers . permission + " , " + newusers . email + " , " +
407             newusers . password + " , " + newusers . name + " , " +
408             newusers . surname + " , " + newusers . otche + " , " + newusers . vk +
409             " , " + newusers . skype + " , " + newusers . telegram + " , " +
410             newusers . course + " , " + newusers . type + " ) " );
411         connection . query ( "UPDATE users SET user_university = (SELECT * FROM
412             university WHERE university_name = " + newusers . university + " )
413             WHERE email = " + newusers . email + " " );
414         connection . query ( "UPDATE users SET user_faculty = (SELECT faculty_name
415             FROM faculty WHERE faculty_name = " + newusers . faculty + " ) WHERE
416             email = " + newusers . email + " " );
417         connection . query ( "UPDATE users SET user_direction = (SELECT
418             direction_name FROM direction WHERE direction_name = " +
419             newusers . direction + " ) WHERE email = " + newusers . email + " " );
420     }
421     else { //если работодатель
422         newusers = { //определяем поля работодателя
423             permission : req . body . permission ,
424             email : req . body . email ,
425             password : req . body . password ,
426             name : req . body . name ,
427             surname : req . body . surname ,
428             otche : req . body . otche ,
429             country : req . body . country , //связанное поле

```

```

418     city: req.body.city, //связанное поле
419     vk: req.body.vk,
420     skype: req.body.skype,
421     telegram: req.body.telegram
422   };
423   connection.query("INSERT INTO users (permission, email, password, name,
424     surname, patronymic, id_vk, skype, telegram) VALUES('" +
425     newusers.permission + "', '" + newusers.email + "', '" +
426     newusers.password + "', '" + newusers.name + "', '" +
427     newusers.surname + "', '" + newusers.otche + "', '" + newusers.vk +
428     "', '" + newusers.skype + "', '" + newusers.telegram + "')");
429 }
430 var status = { //cmamyc
431   send: true
432 }
433 if (newusers.country != ''){
434   connection.query("UPDATE users SET user_country = (SELECT * FROM country
435     WHERE country_name = '" + newusers.country + "') WHERE email = '" +
436     newusers.email + "'");
437   if (newusers.city != ''){
438     connection.query("UPDATE users SET user_city = (SELECT city_name FROM
439       city WHERE city_name = '" + newusers.city + "') WHERE email = '" +
440         newusers.email + "'");
441   }
442 }
443 res.send(status); //отправка статуса
444 });
445
446 app.post('/email', function(req, res){ //ответ на запрос проверки email
447   var email = { //объект email
448     found: false //значение поиска
449   };
450   connection.query('SELECT * FROM users WHERE email = "' + req.body.email +
451     '"', function(err, docs){ // поиск в базе данных пользователей введенного
452     email.found
453     if (err){ //выход если ошибка
454       console.log(err);
455     }
456     else if (docs[0]){ //если найдено совпадение
457       email.found = true; //присваиваем положительный статус
458     }
459     else{ //иначе
460       email.found = false; //отрицательный
461     }
462     res.send(email); //отправка статуса
463   });
464 });

```

```

455 app.post('/sendmail', function(req, res){//ответ на запрос восстановления пароля
456     var status = {//статус восстановления
457         send: '',
458     };
459
460     var transporter = nodemailer.createTransport({//настройки обработчика
461         отправки писем
462         service: "Gmail",//сервис
463         host: 'smtp.ethereal.email',//хост
464         port: 587,//порт
465         secure: false,
466         auth: {//автентификация отправителя
467             user: "CarmaWorkServices@gmail.com",//логин
468             pass: "Stark009s"//пароль
469         }
470     });
471
472     connection.query('SELECT * FROM users WHERE email = "' + req.body.email +
473         '"', function(err, docs){//поиск в базе данных зарегистрированных
474         пользователей
475         if(err){//вывод если ошибка
476             console.log(err);
477             status.send = false;
478         }
479         else if(docs[0]){//если найдено совпадение
480             var mail = {//конструктор письма
481                 from: "CarmaWork <from@gmail.com>",//отправитель
482                 to: req.body.email,//адресат
483                 subject: "Восстановление пароля",//тема
484                 text: "Ваш Email: " + docs.email + " пароль: " +
485                     docs.password,//текст
486                 html: "<b>Ваш Email: </b>" + docs[0].email + "<b> пароль: </b>" +
487                     docs[0].password//html текст
488             };
489
490             transporter.sendMail(mail, function(error, info){//отправка
491                 if(error){//вывод если ошибка
492                     console.log(error);
493                     };//вывод в консоль статуса отправки
494                     console.log('Message sent: %s', info.messageId);
495                     console.log('Preview URL: %s',
496                         nodemailer.getTestMessageUrl(info));
497                     transporter.close();//закрытие потока отправки
498                 });
499                 status.send = true;//присваиваем положительный статус отправки
500             }
501             else{//иначе
502                 status.send = false;//отрицательный статус
503             }
504         }
505     });

```

```

497      }
498      res.send(status); //отправка статуса
499  });
500 );
501
502 app.post('/sendmailcode', function(req, res){ //ответ на запрос отправки письма
503   // кодом для регистрации
504   var status = { //статус
505     send: '',
506   };
507
508   var transporter = nodemailer.createTransport({ //настройки обработчика
509     // отправки писем
510     service: "Gmail", //сервис
511     host: 'smtp.ethereal.email', //хост
512     port: 587, //порт
513     secure: false,
514     auth: { //авторизация отправителя
515       user: "CarmaWorkServices@gmail.com", //логин
516       pass: "Stark009s" //пароль
517     }
518   });
519
520   const crypto = require('crypto'); //модуль шифрования
521   crypto.randomBytes(8, (err, buf) => { //функция генерирования 8 битного кода
522     if (err){ //выход если ошибка
523       console.log(err);
524       status.send = false; //отрицательный статус
525     }
526     else{ //иначе
527       connection.query("DELETE FROM code WHERE email = '" + req.body.email +
528         "'"); //удаление старого кода
529       connection.query("INSERT INTO code (code_name, email) VALUES ('" +
530         buf.toString('hex') + "', '" + req.body.email + "')"); //вставка нового кода
531
532       var mail = { //конструктор письма
533         from: "CarmaWork <from@gmail.com>", //отправитель
534         to: req.body.email, //адресат
535         subject: "Код подтверждения регистрации", //тема
536         text: "Ваш код подтверждения регистрации: " +
537           buf.toString('hex'), //текст
538         html: "<b>Ваш код подтверждения регистрации: </b>" +
539           buf.toString('hex') //html текст
540       };
541
542       transporter.sendMail(mail, function(error, info){ //отправка
543         if(error){ //выход если ошибка
544           console.log(error);
545         } //выход в консоль статуса отправки

```

```

539         console.log('Message sent: %s', info.messageId);
540         console.log('Preview URL: %s',
541                     nodemailer.getTestMessageUrl(info));
542     transporter.close(); //закрытие потока отправки
543   });
544   status.send = true; //положительный статус
545   res.send(status); //отправка статуса
546 });
547 });
548
549 app.post('/checkcode', function(req, res){//ответ на запрос проверки кода
550   var status = { //статус
551     send: '' ,
552   };
553   connection.query('SELECT * FROM code WHERE code_name = "' + req.body.code +
554     '"', function(err, docs){ //поиск в базе данных пользователей введенного
555     email
556     if(err){ //выход если ошибка
557       console.log(err);
558       status.send = false;
559     }
560     else if(docs[0]){ //если найдено совпадение
561       status.send = true; //устанавливаем положительный статус
562       connection.query("DELETE FROM code WHERE code_name = '" +
563         req.body.code + "'");
564     }
565     else{ //иначе
566       status.send = false; //отрицательный статус
567     }
568   });
569 }
570
571 app.post('/permission', function(req, res){ //ответ на запрос информации о типе
572   учетной записи пользователя
573   var status = { //информация о пользователе
574     permission: '', //тип учетной записи
575     university: '', //университет (для студента)
576     faculty: '', //факультет (для студента)
577     email: '' ,
578   };
579
580   connection.query('SELECT * FROM sessions WHERE sid = "' + req.sessionID +
581     '"', function(err, docs){
582     if(err){
583       console.log(err);
584     }

```

```

581     else if (docs[0]) {
582         status.permission = docs[0].permission;
583         status.email = docs[0].sessions_email;
584     }
585     if (status.permission == 1){//если студент
586         connection.query('SELECT * FROM users WHERE email = "' + status.email
587                         + '"', function(err, docs){//поиск студента в базе данных по данным
588                         if (err){//вывод если ошибка
589                             console.log(err);
590                         }
591                         else if (docs[0]){//если найдено совпадение
592                             status.university = docs[0].user_university;//устанавливаем
593                             status.faculty = docs[0].user_faculty;//у факультета
594                         }
595                         res.send(status); //отправляем данные студента
596                     });
597     }
598     else{
599         res.send(status); //отправляем данные работодателя
600     });
601
602
603 });
604
605 app.post('/load', function(req, res){//ответ на запрос карточек
606     var load = {//полученные данные
607         course: req.body.course,
608         type: req.body.type,
609         university: req.body.university,
610         faculty: req.body.faculty,
611         direction: req.body.direction
612     };
613     var sql = "SELECT * FROM users WHERE permission = '1'";
614     var filtr = {//фильтр
615         permission: '1'//поиск только по студентам
616     };
617     if (load.course != ''){//если получен не пустой курс
618         filtr.course = load.course;//устанавливаем фильтр по нему
619         sql += " AND kurs = '" + load.course + "'";
620     }
621     if (load.type != ''){//если получена не пустая форма обучения
622         filtr.type = load.type;//устанавливаем фильтр по форме обучения
623         sql += " AND form = '" + load.type + "'";
624     }
625     if (load.university != ''){//если получен не пустой университет

```

```

626     filtr.university = load.university; //устанавливаем фильтр по университету
627     sql += " AND user_university = '" + load.university + "'";
628     if(load.faculty != ''){//если получен не пустой факультет
629         filtr.faculty = load.faculty; //устанавливаем фильтр по факультету
630         sql += " AND user_faculty = '" + load.faculty + "'";
631     if(load.direction != ''){//если получено не пустое направление
632         filtr.direction = load.direction; //устанавливаем фильтр по
633             направлению
634             sql += " AND user_direction = '" + load.direction + "'";
635     }
636 }
637 connection.query(sql, function(err, docs){
638     if(err){//вывод если ошибка
639         console.log(err);
640     }
641     else if(docs.size === 0){//если найдено совпадение
642         var users = [];//массив объектов студентов
643         docs.forEach(function(element){//проход по найденным студентам
644             var user = {};//объект студент
645             user.name = element.name;//присваиваем имя
646             user.surname = element.surname;//фамилию
647             user.otche = element.patronymic;//отчество
648             user.email = element.email;//email
649             user.university = element.user_university;//университет
650             user.faculty = element.user_faculty;//факультет
651             user.direction = element.user_direction;//направление
652             user.type = element.form;//форму обучения
653             user.course = element.kurs;//курс
654             user.vk = element.id_vk;//vk
655             user.telegram = element.telegram;//telegram
656             user.skype = element.skype;//skype
657             users.push(user);//добавляем в массив
658         });
659         res.send(users); //отправляем данные
660     }
661 });
662 });

```

ПРИЛОЖЕНИЕ В

ER - диаграмма

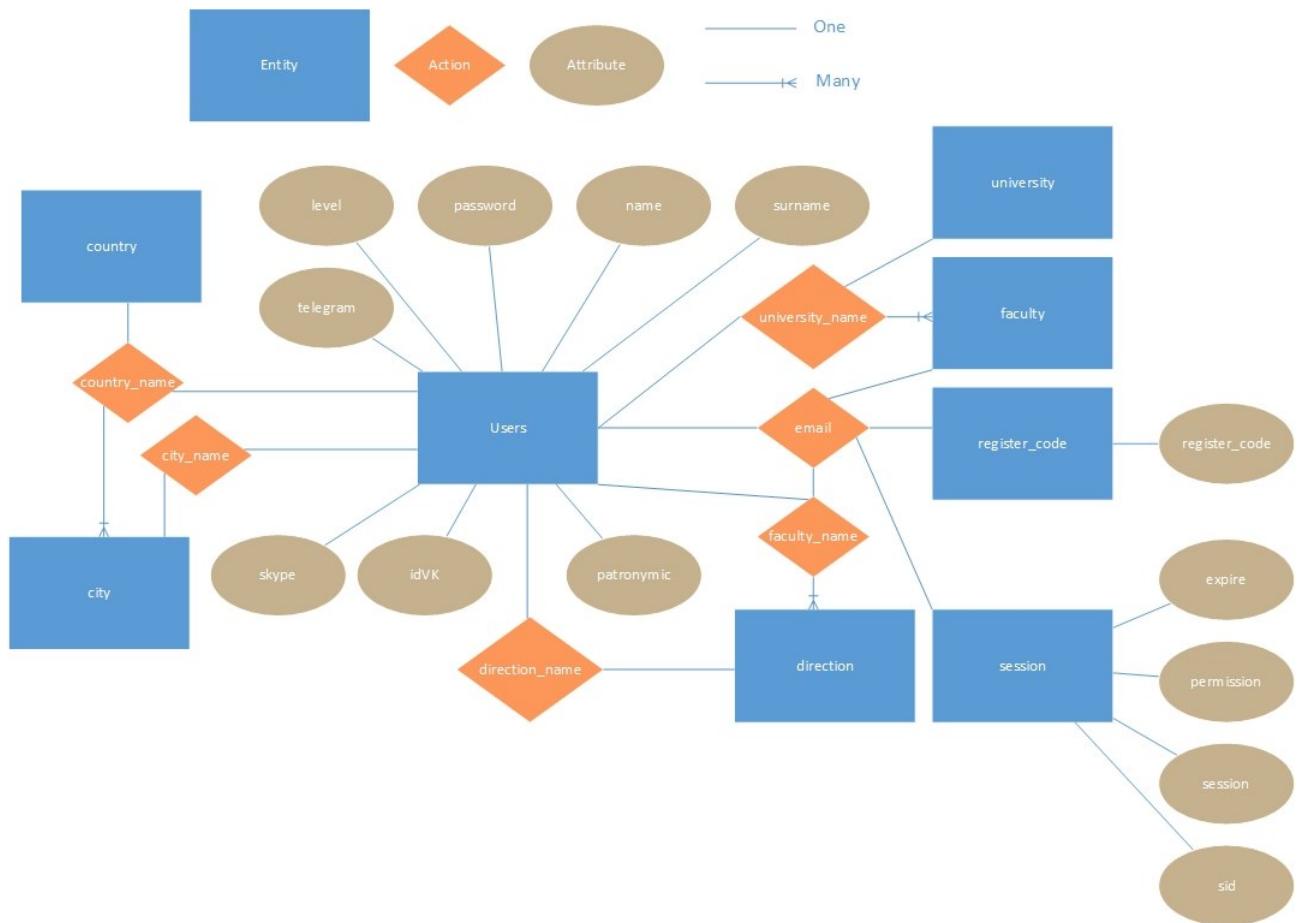


Рисунок 19 – ER-диаграмма