Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

К защите допустить:

Заведующий кафедрой ПИКС

_____И.Н. Цырельчук

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

БАЗА ДАННЫХ ОТДЕЛА КАДРОВ МЕДИЦИНСКОГО УЧРЕЖДЕНИЯ

БГУИР КП 1-39 03 02 010 ПЗ

Студент А.В. Коваль

Руководитель И.Н. Богатко

Минск 2015

РЕФЕРАТ

ГУИР 313801 010 ПЗ

Коваль, А.В. База данных отдела кадров медицинского учреждения : пояснительная записка к курсовому проекту / А. В. Коваль. – Минск : БГУИР, 2015. – п.з. – 40 с., чертежей (плакатов) – 3 л. формата А1.

Пояснительная записка 40 с., 6 рис., 4 табл., 18 источников, 4 приложения

БАЗА ДАННЫХ ОТДЕЛА КАДРОВ МЕДИЦИНСКОГО УЧРЕЖДЕНИЯ

Цель проектирования: разработка базы данных и пользовательского интерфейса для доступа к ней.

Предмет проектирования: непосредственно разрабатываемая база данных. *Объект проектирования* – это реляционная система управления базами данных MySQL.

Актуальность работы: Разработка базы данных объясняется необходимостью автоматизации процесса учета и обработки данных предметной области с целью упрощения и ускорения производственного процесса.

Методология проектирования: Для проведения проектирования, в качестве основополагающего, используется метод «нисходящего» проектирования. При «нисходящем» проектировании осуществляется структурное проектирование сверху—вниз. Такой процесс называют анализом – происходит изучение целого (описания предметной области), затем разделение целого на составные части и далее следует последовательное изучение этих частей.

Анализ предметной области позволяет выявить информационные потребности пользователей. Разработка инфологической модели предметной области также является неотъемлемой частью программной реализации базы данных. Для удобства пользователя, ему необходимо предоставить интуитивно понятный доступ к разрабатываемой базе.

Результаты работы: В процессе изучения объекта проектирования была разработана законченная база данных отдела кадров медицинского учреждения, предназначенная для пользования различными категориями пользователей. Также был разработан пользовательский интерфейс, с помощью которого пользователи базы смогут работать с ней и производить необходимые манипуляции с данными.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Работа отдела кадров медицинского учреждения неразрывно связана с обработкой информации. До недавнего времени в качестве носителей информации использовались бумажные картотеки и архивы. Их недостатком была необходимость вручную вносить, сверять, обновлять и удалять данные, а также невысокая надежность и отсутствие возможности быстрого резервирования и восстановления утерянных данных. В качестве альтернативы появились цифровые базы данных на базе персональных компьютеров, а также различные системы по управлению этими базами данных.

К основным преимуществам цифровых баз данных относят:

– простота использования настроенной базы данных;
– обеспечение целостности данных;
– наличие механизмов резервирования данных;
– наличие механизмов восстановления данных;
– возможность организации защищенного доступа к данным;
– возможность разграничения полномочий доступа к данным для различных категорий пользователей;
– широкий выбор бесплатных и условно-бесплатных решений на рынке;
– широкий спектр задач, решаемый современными базами данных;
– обеспечение возможности удаленного доступа к базе данных;
– обеспечение возможности множественного доступа к базе данных.

Среди прочих доступных решений наибольшее распространение получили реляционные базы данных, поскольку они позволяют хранить данные в связанных между собой таблицах. Распространенность именно этого типа баз данных обусловлена тем, что данные, которыми оперирует отдел кадров предприятия, характеризуются высокой степенью формализации и унификации – это позволяет хранить информацию в таблицах с заранее определенным набором полей и заранее определенными связями между этими таблицами.

Для работы с базами данных используется специальное программное обеспечение – системы управления базами данных. Системы управления базами данных (СУБД) предоставляют пользователю программно-аппаратные средства для доступа к базе данных; набор механизмов для внесения, чтения, изменения и удаления информации, содержащейся в базе данных; инструменты для организации структуры базы данных, набор служебных сервисов, необходимых для ведения лога изменений, резервирования, репликации, возможности обеспечения многопользовательского доступа к базе данных и прочих инструментов, наличие которых определяется

конкретной реализацией СУБД. На данный момент наиболее популярны следующие системы, поддерживающие работу с реляционными базами данных: MySQL, PostgreSQL, MSSQL Server, Oracle database.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ЕЁ ФОРМАЛИЗАЦИЯ ДЛЯ ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ

1.1 Описание предметной области

Предметной областью данного курсового проекта является область деятельности отдела кадров медицинского учреждения. Отдел кадров как структурное подразделение оперирует данными о сотрудниках учреждения, об их принадлежности к различным отделам и организационным структурам, документах, необходимых для учета, сбора и обработки информации о сотрудниках (например, паспортные данные, сведения о контрактах и т.д.). Также в базе данных отдела кадров хранятся справочные данные о возможных графиках работы, тарифных ставках и окладах для специалистов различной квалификации, и т.д. Характерной чертой, отличающей отдел кадров медицинского учреждения от отдела кадров любого другого учреждения, является наличие сведений о медицинских лицензиях и датах переаттестации/подтверждения квалификации медицинского персонала, а также специфическая структура отделов (больничных отделений).

Процесс работы отдела кадров медицинского учреждения включает в себя следующие этапы:

1 Заявление медицинского учреждения о потребности в сотруднике определенной квалификации на определенную должность. В заявлении помимо основной информации могут быть также указаны дополнительные сведения, такие как информация о зарплате, организации рабочего времени и прочих особенностях.

2 Подача первичных данных о кандидате на вакансию в отдел кадров учреждения и внесение этих данных в сводную ведомость кандидатов.

3 По итогам собеседования/конкурса из множества кандидатов на должность выбирается один. Этот кандидат зачисляется в штат учреждения. Полные сведения о сотруднике вносятся в базу данных.

4 В случае изменения данных сотрудника записи о сотруднике в базе данных актуализируются.

5 В случае увольнения сотрудника соответствующие записи удаляются из базы данных или перемещаются в архив.

В базе данных должны быть отражены следующие сведения:

– сведения о банковских счетах сотрудников;

– сведения о потенциальных кандидатах на определенные должности.;

– контактные данные сотрудников;

– сведения о контрактах;

– сведения об отделах учреждения;

– сведения об образовании сотрудника;

– общая информация о сотрудниках;

– сведения о медицинских картах сотрудников;

– сведения о воинском учете сотрудников;

– паспортные данные сотрудников;

– сведения о должностях;

– сведения о номерах рабочих телефонов, которые принадлежат сотрудникам с определенными должностями;

– сведения о кабинетах (корпусах), в которых располагаются рабочие места сотрудников определенных должностей;

– сведения о льготах, которые имеют сотрудники, описание семейного положения и данных о детях сотрудников;

– сведения о трудовых книжках сотрудников.

Также база данных должна включать справочные сведения о возможных уровнях образования, ученых степенях, воинских званиях, типах оплаты труда сотрудника и прочих справочных данных.

1.2 Анализ информационных потребностей пользователей и предварительное описание запросов

В рамках информационной системы «отдел кадров медицинского учреждения» предполагается, что пользователи имеют следующие информационные потребности:

– Поиск данных о сотруднике (ФИО) по занимаемой должности;

– Отображение списка сотрудников отдела;

– Отображение списка сотрудников с контрактом, истекающим в этом месяце/году;

– Отображение данных о зарплате и расчетном счете сотрудника;

– Отображение данных о медицинских лицензиях;

– Отображение данных о медицинских сотрудниках, подлежащих аккредитации в текущем году;

– Отображение сведений о кандидатах, претендующих на определенную должность;

– Отображение данных о занимаемой сотрудником должности (принадлежности к определенному отделу) на основе табельного номера сотрудника;

– Отображение сведений о руководителе отдела и его контактных данных;

– Поиск сотрудника на основе паспортных данных;

– Отображение сведений о сотрудниках, состоящих на воинском учете;

– Поиск специалистов с определенной квалификацией;

– Поиск специалистов с выбранным уровнем образования (ученой степенью);

– Поиск сотрудников определенной квалификации с определенным стажем работы.

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

С точки зрения предметной области, в базе данных должны быть учтены следующие ограничения:

– Должен существовать уникальный идентификатор сотрудника (например, табельный номер);

– База данных должна обладать механизмами защиты от несанкционированного доступа, т.к. в ней хранится конфиденциальная информация (например, паспортные данные и сведения о банковских счетах);

– При обновлении или удалении информации из базы данных, должна быть обеспечена целостность операции, т.е. идентификаторы и связанные с ними строки должны обновляться/удаляться каскадно;

– Данные не должны быть логически противоречивы (например, дата заключения контракта не должна быть позже даты истечения контракта);

– В базе данных необходимо определить роли пользователей с различными уровнями доступа и правами на чтение/изменение/добавление/удаление информации;

– Для пользователей базы данных необходимо определить уровни доступа, соответствующие полномочиям пользователя.

1.4 Постановка решаемой задачи

Постановка решаемой задачи сводится к следующему:

1 Анализ предметной области и её формализация для проектирования базы данных.

2 Проектирование базы данных для основного вида деятельности рассматриваемой предметной области.

3 Применение разработанной базы данных.

2 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ ДЛЯ ОСНОВНОГО ВИДА ДЕЯТЕЛЬНОСТИ РАССМАТРИВАЕМОЙ ПРЕДМЕТНОЙ ОБЛАСТИ

2.1 Разработка инфологической модели предметной области базы данных

Исходя из темы проекта курсового проектирования, необходимо спроектировать базу данных отдела кадров медицинского учреждения.

В рамках анализа предметной области были рассмотрены две составляющие: предметная область работы отдела кадров и предметная область работы медицинского учреждения.

Предметная область работы отдела кадров характеризуется обработкой данных о сотрудниках учреждения, кандидатах на работу, а также информацией, касающейся характера и условий труда сотрудников. В рамках исследования предметной области было определено, что потенциальные сущности – фрагменты информации о сотруднике, характеризующие его с определенной стороны.

Предметная область работы медицинского учреждения определяет наличие специфических данных, характеризующих сотрудника.

Таким образом в рамках составления инфологической модели предметной области можно выделить следующие сущности:

Таблица 2.1 – Сущности

Сущность	Поля, описывающие сущность	Описание сущности
Банковская информация	Табельный номер, расчетный счет	Сущность описывает данные, необходимые для проведения банковских расчетов с сотрудником
Информация о кандидатах	Вакансия, ФИО, возраст, пол, уровень образования, специальность, контактные данные	Сущность описывает данные, характеризующие кандидатов на вакантную должность
Контактная информация	Табельный номер, номера телефонов, адрес проживания, адрес электронной почты	Сущность описывает контактные данные сотрудника
Информация о контрактах	Номер контракта, зарплата, дата подписания договора, дата истечения договора	Сущность содержит информацию о договорах, заключенных с сотрудником
Информация об отделах	Идентификатор отдела, название отдела, табельный номер руководителя отдела	Сущность содержит информацию об отделах учреждения

Продолжение таблицы 2.1

Сущность	Поля, описывающие сущность	Описание сущности
Информация об образовании	Серия и номер документа об образовании, год окончания УО, наименование УО, уровень образования	Сущность содержит информацию, отражающую сведения об образовании сотрудника
Основная информация о сотруднике	Табельный номер, ФИО, пол, возраст	Сущность описывает основные сведения о сотруднике
Сводная таблица информации	Табельный номер, номер паспорта, номер договора, номер мед.карты, номер военного билета, должность, серия и номер документа об образовании, серия и номер трудовой книжки, номер медицинской лицензии	Агрегированная сущность для обеспечения связности данных о сотруднике
Медицинская информация	Номер медицинской карты, наличие инвалидности, группа инвалидности, примечание	Сущность отражает данные о медицинской документации сотрудника
Информация о медицинских лицензиях	Номер медицинской лицензии, дата последней аккредитации, дата следующей аккредитации	Сущность отражает данные о состоянии медицинских лицензий
Информация о воинской повинности	Номер военного билета, воинское звание, номер в/ч, присвоенная квалификация	Сущность описывает данные о сотрудниках, состоящих на воинском учете
Паспортные данные	Идентификационный номер паспорта, номер паспорта, серия, дата рождения, адрес прописки, гражданство	Сущность содержит информацию о паспортных сведениях сотрудника
Информация о должности	Идентификтор должности, наименование должности, требуемая квалификация, требуемый уровень образования, минимальный оклад	Сущность содержит сведения обо всех должностях, имеющихся в учреждении
Информация о трудовых книжках	Серия и номер трудовой книжки, дата выдачи, образование, специальность, серия, номер, дата выдачи вкладыша	Сущность содержит сведения о состоянии трудовых книжек сотрудников

Также в базе данных имеются вспомогательные сущности, не имеющие непосредственного отношения к данным о сотрудниках, но которые предоставляют справочную информацию, необходимую для формирования основных сущностей.

Перечень вспомогательных сущностей:

Таблица 2.2 – Сущности

Сущность	Поля, описывающие сущность
Справочник уровней образования	Идентификатор уровня образования, описание
Справочник степеней образования	Идентификатор степени образования, описание
Справочник воинских званий	Идентификатор звания, описание
Телефонный справочник учреждения	Идентификатор должности, номер телефона
Справочник местонахождения кабинетов сотрудников	Идентификатор должности, номер корпуса, номер кабинета

Поскольку, в силу специфики предметной области, не представляется возможным произвести именование связей между сущностями, то была принята модель отношений “is-a” (является) для всех отношений составленной модели.

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

В таблице 2.3 представлена общая информация об используемых в разрабатываемой базе данных типах данных. [9-11].

Таблица 2.3 – Типы данных, используемые в базе

Тип данных	Описание	Атрибуты	Обоснование
INT (UNSIGNED)	Целочисленный тип данных. Диапазон: от 0 до 4294967295	Табельный номер, номер договора, номер медицинской лицензии, зарплата, номер военного билета, номер документа об образовании	Поля данного типа всегда описываются численным значением и могут иметь достаточно большие значения

Продолжение таблицы 2.3

VARCHAR	Строковый тип данных	Имя, фамилия, отчество, наименования отделов, должностей, специальностей, званий и пр., адрес, примечания, описание уровня и степени образования, квалификации	Данные атрибуты могут содержать строки переменной длины.
CHAR	Строковый тип данных фиксированной длины	Идентификационный номер паспорта, серия паспорта	Данные типы имеют фиксированную длину и представлены различными символами
SMALLINT (UNSIGNED)	Целочисленный тип данных. Диапазон от 0 до 65535	Идентификатор должности, идентификатор отдела	Поля данного типа всегда принимают относительно небольшие целочисленные значения
TINYINT	Целочисленный тип данных. Диапазон от 0 до 255	Идентификатор звания, уровня образования, степени образования	Поля данного типа всегда принимают небольшие целочисленные значения
ENUM(‘м’,’ж’)	Перечисление. Может принимать значения из определенного набора	Пол	Поля данного типа всегда принадлежат одному из перечисленных значений
DATE	Дата. Формат: *YYYY-MM-DD* Разделителем могут быть любые символы кроме цифр	Дата рождения, дата последней/следующей аккредитации, дата заключения/истечения договора	Позволяет хранить дату в строго типизированном формате
BLOB	Двоичный объект	Номера банковских счетов	Позволяет хранить данные в зашифрованном виде
BOOL	Истина/Ложь	Наличие инвалидности	Принимает только значения истина/ложь

Примечание: у типов данных используются следующие спецификаторы:

<Тип данных> (M) – числовые типы данных содержат M разрядов, строковые – M символов;

<Числовой тип данных> (M) ZEROFILL – свободные позиции слева будут заполняться нулями до разряда M.

UNSIGNED – числовой тип данных без знака (положительный). Применён ко всем числовым типам данных;

NOT NULL – гарантирует, что кортеж не добавится в таблицу, если не заполнен какой-либо его атрибут. Применён ко всем атрибутам кроме некоторых имен, фамилий, ID;

AUTO_INCREMENT – позволяет автоматически заполнять поле атрибута значением на единицу больше максимального значения в нём. Применён к атрибутам, которые являют собой идентификатор отношения;

UNIQUE – гарантирует уникальность значений в атрибуте. Применён к атрибутам, которым необходима уникальность значений;

2.3 Проектирование запросов к базе данных

Информация об информационных потребностях потенциальных пользователей предоставлена в разделе 1.2. Для реализации некоторых из этих потребностей были написаны следующие запросы SQL:

Таблица 2.4 – Реализация запросов к базе

Запрос в словесной форме	Программная реализация запроса
1	2
Поиск данных о сотруднике (ФИО) по занимаемой должности	`select general_info.last_name, general_info.first_name, general_info.middle_name, post_info.post_descr from main_info inner join general_info on main_info.personnel_number = general_info.personnel_number inner join post_info on main_info.post_id = post_info.post_id where post_descr=@argument;`

Продолжение таблицы 2.4

Отображение списка сотрудников отдела	`select dept_info.dept_descr, post_info.post_descr, general_info.last_name, general_info.first_name, general_info.middle_name`
	`  from main_info`
	`  inner join general_info`
	`  on main_info.personnel_number = general_info.personnel_number`
	`  inner join post_info`
	`  on main_info.post_id = post_info.post_id`
	`  inner join dept_info`
	`  on post_info.post_dept_id=dept_info.dept_id`
	`  where dept_id=@argument;`

Отображение списка сотрудников с контрактом, истекающим в этом месяце/году	`select contract_info.contract_end_date, dept_info.dept_descr, post_info.post_descr, general_info.last_name, general_info.first_name, general_info.middle_name, general_info.personnel_number`
	`  from main_info`
	`  inner join general_info on main_info.personnel_number = general_info.personnel_number`
	`  inner join post_info on main_info.post_id = post_info.post_id`
	`  inner join dept_info on post_info.post_dept_id=dept_info.dept_id`
	`  inner join contract_info on main_info.contract_number=contract_info.contract_number where year(contract_info.contract_end_date)= year(curdate());`

Отображение данных о медицинских лицензиях	`select general.last_name, general.first_name, general.middle_name,`
	`  main.medical_license_number,`
	`  date_format(medical_license_info.last_accreditation_date, '%d-%m-%Y') as last_accreditation_date,`
	`  date_format(medical_license_info.next_accreditation_date, '%d-%m-%Y') as next_accreditation_date`
	`  from main_info as main`
	`  inner join general_info as general`
	`  on general.personnel_number = main.personnel_number`
	`  inner join medical_license_info`
	`  on main.medical_license_number = medical_license_info.medical_license_number;`

Продолжение таблицы 2.4

Обновление информации о сотруднике	create procedure cl_updateUser
	(in _personnel_number int unsigned,in _last_name varchar(45), in _first_name varchar(45), in _middle_name varchar(45), in _age tinyint, in _post_id smallint unsigned, in _medical_license_number int unsigned, in _next_accreditation_date date)
	begin
	declare _medical_license_number int;
	select medical_license_number into _medical_license_number from main_info where personnel_number=_personnel_number;
	if (_last_name is not null) then update general_info set last_name = _last_name where personnel_number = _personnel_number;
	end if;
	if (_first_name is not null) then update general_info set first_name = _first_name where personnel_number = _personnel_number;
	end if;
	if (_middle_name is not null) then update general_info set middle_name = _middle_name where personnel_number = _personnel_number;
	end if;
	if (_age is not null) then update general_info set age = _age where personnel_number = _personnel_number;
	end if;
	if (_post_id is not null) then update main_info set post_id = _post_id where personnel_number = _personnel_number;
	end if;
	if (_medical_license_number is not null) then
	update medical_license_info set medical_license_number = _medical_license_number
	where medical_license_number in
	(select medical_license_number from main_info where personnel_number = _personnel_number);
	end if;
	if (_next_accreditation_date is not null) then
	update medical_license_info set next_accreditation_date = _next_accreditation_date where medical_license_number in (select medical_license_number from main_info where personnel_number = _personnel_number);
	end if;
	end;

2.4 Программная реализация и документирование базы данных

Реализация описанной базы данных и всех запросов к ней была осуществлена на языке SQL СУБД MySQL. В качестве прикладных программ для реализации схемы базы данных и создания скриптов использовались MySQL Workbench и MySQL 5.6 Command Line.

Программа MySQL использовалась для создания схемы базы данных, определения таблиц, установления связей между отношениями и создания хранимых процедур. Процесс создания схемы базы данных представляет собой следующую последовательность действий:

1 Создание т.н. модели – файла формата *.mwb, который будет содержать в себе данные о структуре проектируемой БД.

2 Создание таблиц внутри созданной модели возможно путем создания добавления таблиц с помощью элементов графического интерфейса на панели управления моделью, с помощью добавления таблиц на ER-диаграмму модели, с помощью инструмента scripting shell.

3 Создание связей между отношениями происходит при помощи определения ключей в свойствах создаваемой таблицы.

После создания схемы данных необходимо сгенерировать так называемый SQL CREATE Script, который будет использован непосредственно на сервере базы данных. Созданный скрипт возможно запустить как при помощи MySQL Command Line Client, так и при помощи утилиты импорта данных, доступной в настройках MySQL сервера.

Для документирования базы данных были разработаны следующие диаграммы, которые представлены в приложении Г: диаграмма «сущность-связь», диаграмма бизнес-процессов. Данные диаграммы были реализованы в программах MS Visio, CorelDraw.

3 ПРИМЕНЕНИЕ РАЗРАБОТАННОЙ БАЗЫ ДАННЫХ

3.1 Руководство пользователя

Для обеспечения пользовательского доступа к базе данных было разработано приложение с графическим интерфейсом. Приложение было реализовано с использованием клиент-серверной архитектуры. Для разработки серверной части использовался NodeJS. Для разработки клиентской части приложения использовался фреймворк языка JavaScript AngularJS, а также язык разметки HTML и язык каскадных стилей CSS.

Для начала работы с приложением пользователю необходимо зарегистрироваться в системе, используя уникальный логин и пароль. Регистрационные данные пользователей и доступные им привилегии определяются непосредственно в базе данных.

Форма авторизации представлена на рисунке 3.1:

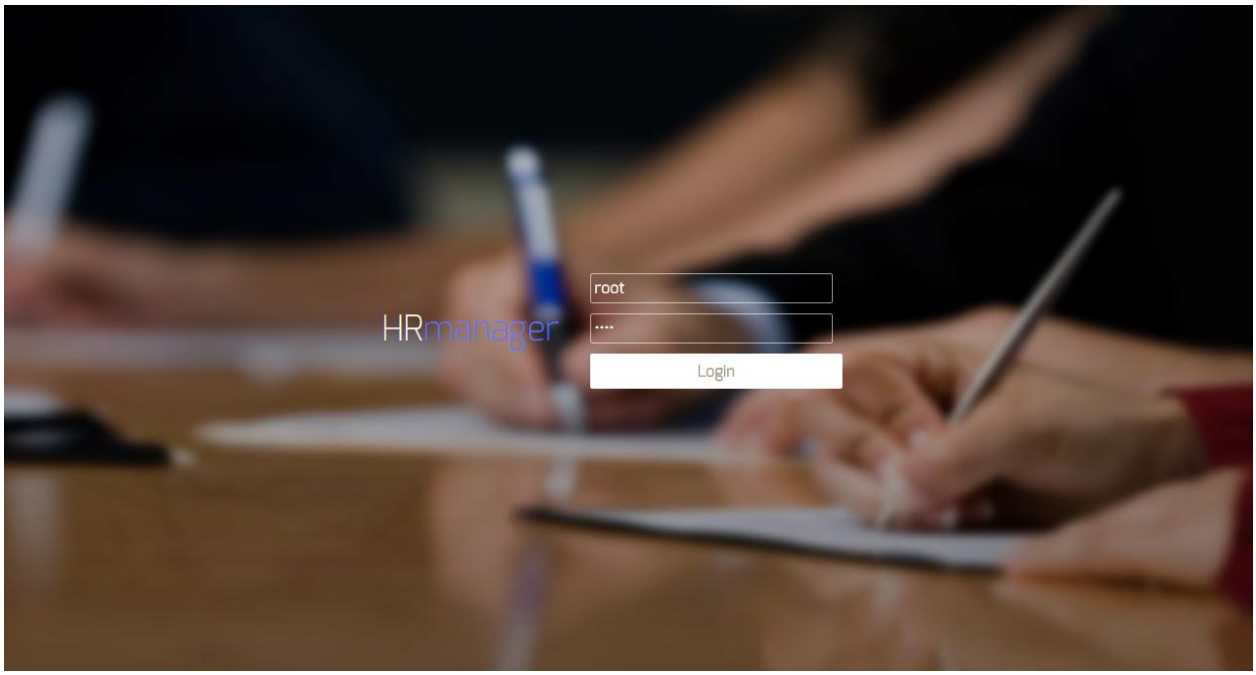

Рисунок 3.1 – Форма авторизации

После авторизации в системе пользователь попадает на страницу, на которой отображены основные сведения о сотрудниках. Переход именно на эту страницу обусловлен тем, что для всех пользователей доступен просмотр основной информации.

В случае, если пользователь ввел неверные данные для авторизации, будет отображено уведомление о том, что введенная комбинация логина и пароля не является валидной.

Интерфейс основной страницы представлен на рисунке 3.2:

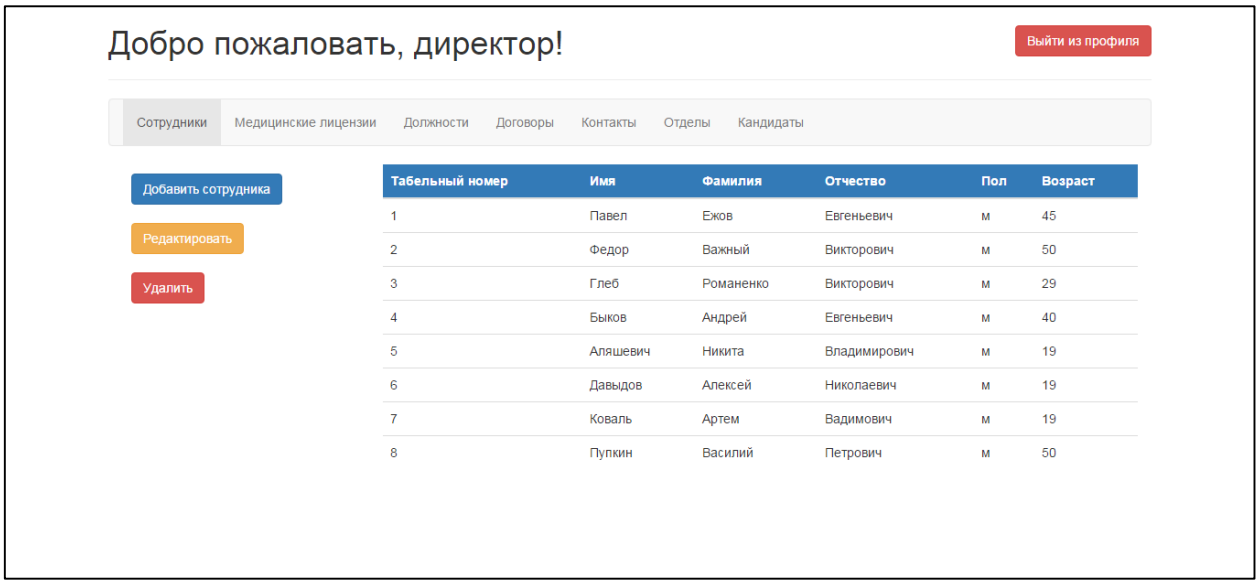

Рисунок 3.2 – Главное окно

С основной страницы возможен переход к просмотру любой из доступных пользователю таблиц. В случае, если пользователь имеет привилегии не только на чтение таблицы, то, согласно доступным привилегиям, в боковой панели будут отображаться кнопки «Добавить», «Редактировать», «Удалить».

При добавлении сотрудника в таблицу появляется всплывающее окно, содержащее поля ввода для параметров, характеризующих сотрудника. Для медицинских сотрудников будут дополнительно доступны поля «Номер лицензии», «Предыдущая аккредитация» и «Следующая аккредитация». Примеры добавления сотрудника медицинской и гражданской должности можно увидеть на рисунках 3.3 и 3.4, соответственно.

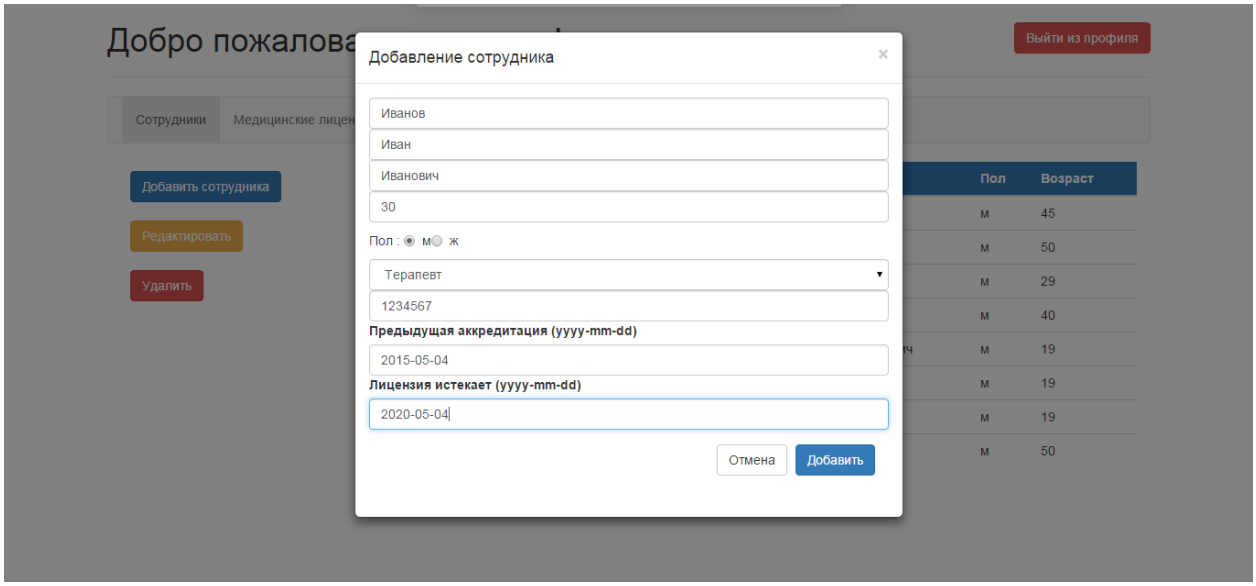

Рисунок 3.3 – Добавление медицинского сотрудника в базу данных

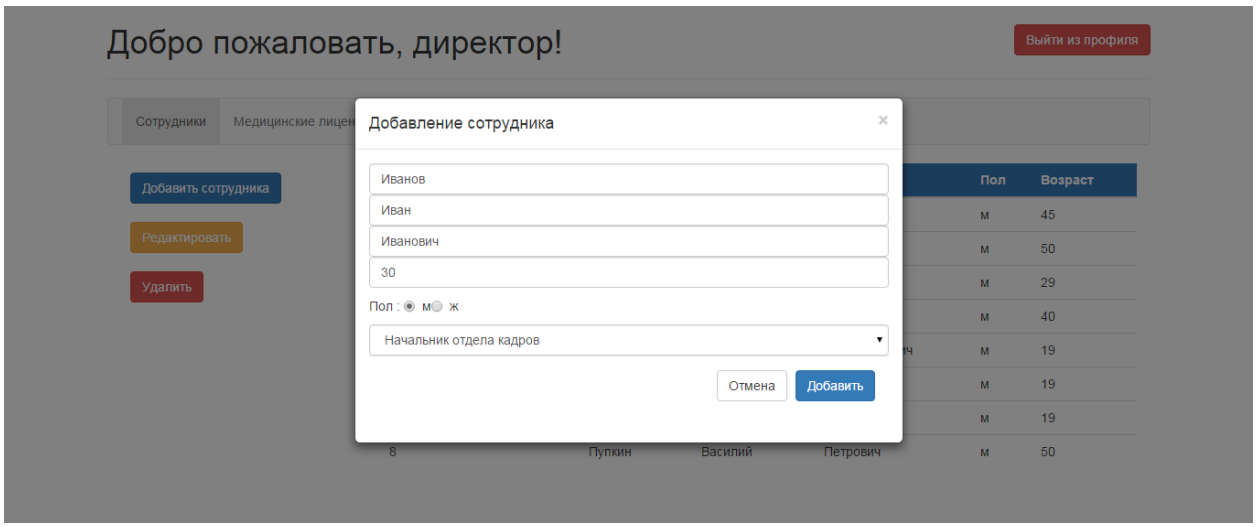

Рисунок 3.4 – Добавление сотрудника гражданской специальности

При нажатии кнопки «Редактировать» будет отображено всплывающее окно, с текущими значениями полей для выбранного сотрудника. Выбрать сотрудника возможно путем клика по соответствующей строке в таблице. В случае, если не выбрана ни одна строка, то пользователю будет отображено уведомление о том, что для операции редактирования необходимо выбрать сотрудника.

При нажатии кнопки «Удалить» будет произведено удаление данных о сотруднике из базы данных. Поскольку данные являются связанными и при удалении записи о сотруднике из одной таблицы, происходит каскадное удаление данных из других таблиц, кнопка «Удалить» доступна только на главной странице.

На других вкладках можно просмотреть специфическую информацию о сотрудниках, например, информацию о состоянии медицинских лицензий (рисунок 3.5), принадлежности сотрудников к определенному отделу, просмотр занимаемых должностей.

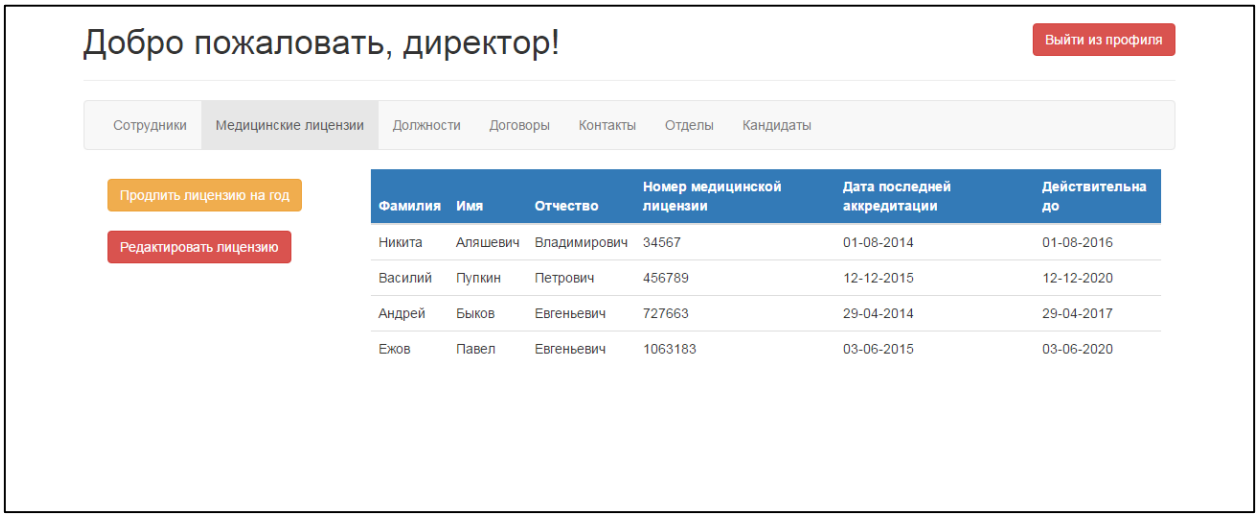

Рисунок 3.5 – Просмотр информации о медицинских лицензиях

На вкладке «Медицинские лицензии» доступны кнопки «Продлить лицензию на год» и «Редактировать лицензию», позволяющие изменять данные для медицинских лицензий сотрудников. Для вызова этих функций необходимо выбрать запись в таблице, иначе будет отображено сообщение об ошибке.

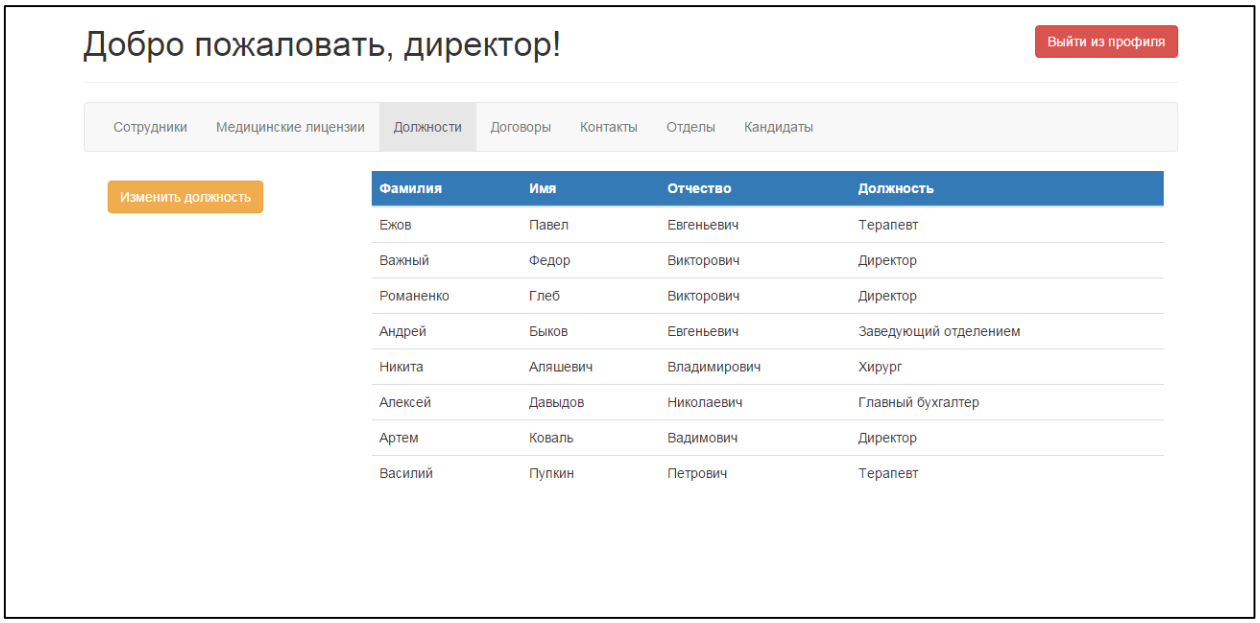

Рисунок 3.6 – Просмотр информации о должностях сотрудников

На вкладке «Должности» пользователь может просмотреть информацию о занимаемых должностях для различных сотрудников. Также имеется возможность изменить должность для выбранного сотрудника.

После окончания работы с приложением необходимо нажать кнопку «Выйти из профиля» в правом верхнем углу экрана.

3.2 Администрирование базы данных

На этапе администрирования необходимо создать пользователей базы данных с соответствующими привилегиями при помощи команды GRANT. Для разрабатываемой базы данных были созданы следующие пользователи: root, secretary, hr, accountant, для которых определены следующие привелегии:

– root – служебный пользователь. Необходим для администрирования базы данных и проверки корректности работы функций. Обладает всеми возможными полномочиями;

– secretary – (секретарь) пользователь, имеющий права на изменение данных во всех таблицах, кроме таблицы bank_info;

– hr – (сотрудник отдела кадров) пользователь, который имеет права на создание записей в таблицах, их изменение, удаление. В таблице bank_info

имеет права только на чтение.

– accountant – (бухгалтер) пользователь, имеющий права на изменение данных в таблицах bank_info и contract_info. Во всех остальных таблицах обладает правами только на чтение.

Листинг создания групп пользователей приведён в приложении В.[17]

3.3 Реализация клиентских запросов

Реализация соединения с базой данных для передачи запросов и получения их результатов на серверном приложении происходит при помощи модуля NodeJS MySQL. Для передачи запросов между клиентом и сервером используется протокол HTTP. Механизм осуществления запросов представляется следующим алгоритмом:

1 При авторизации пользователя в системе происходит инициализация соединения с БД для соответствующего пользователя.

2 На клиенте формируется набор параметров, необходимых для передачи в запрос.

3 Клиентское приложение отправляет серверу HTTP запрос с выбранными параметрами

4 Сервер обрабатывает запрос, извлекает из него запрашиваемые параметры и вызывает метод query() объекта connection модуля MySQL с аргументами в виде наименования хранимой процедуры и требуемыми параметрами.

5 Результат запроса возвращается клиенту в виде HTTP ответа (response)

3.4 Обоснование и реализация механизма обеспечения безопасности и сохранности данных

Для обеспечения сохранности данных при непредвиденных сбоях или умышленных их изменении необходимо производить регулярное резервное копирование базы. Для создания резервной копии базы используется утилита mysqldump, которая записывает копию в .sql файл. Делается данная операция следующей командой:

mysqldump –uroot –prootpass hr > backup.sql;

Для восстановления базы из копии вместо уже существующей производится команда:

mysql –uroot –prootpass hr < backup.sql;

Для защиты от несанкционированного доступа к БД при наличии прямого подключения к серверу, каждый пользователь (root, client, employee, admin) защищены паролями.

ЗАКЛЮЧЕНИЕ

В ходе курсового проектирования была разработана полноценная база данных отдела кадров медицинского учреждения. Она предназначена для использования сотрудниками отдела кадров, бухгалтерии, администрации учреждения. Проектирование осуществлялось с помощью автоматизированного средства проектирования MySQL Wokrbench.

Для удобства использования было разработано веб-приложение, с помощью которого пользователи базы смогут работать с ней и производить необходимые манипуляции с данными. Для работы с базой данных был использован модуль MySQL NodeJS.

Результаты проектирования получили своё отражение в виде различных диаграмм: ER-диаграмма, диаграмма бизнес-процессов. Данные диаграммы были реализованы в программах Visio, CorelDraw.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Хаф, Л. Проектирование информационных систем [Электронный ресурс] / Л. Хаф. – Режим доступа : http://www.infosystem.ru/is/theory/theory_design_is.html#15.

[2] Карпова, И. П. Базы данных: учебное пособие. / И. П. Карпова. – СПб. : Питер, 2013. – 240 стр. (Серия «Учебное пособие»).

[3] Хомоненко, А. Д. Базы данных : учебник для высших учебных заведений / А. Д. Хомоненко, В. М. Цыганков, М. Г. Мальцев; под ред. проф. А. Д. Хомоненко. – 6-е изд., доп. – СПб. : КОРОНА-Век, 2009. – 736 с.

[4] Голицына, О. Л. Базы данных : учебное пособие / О. Л. Голицына, Н. В. Максимов, И. И. Попов. – 3-е изд., перераб. и доп. – М. : ФОРУМ: ИН-ФРА-М, 2006. – 352 с: ил. – (Высшее образование: бакалавриат).

[5] Ramez Elmasri. Fundamentals of Database Systems / Ramez Elmasri, Shamkant B. Navathe. – the 6-th edition. – Addison-Wesley, 2010. – 1172 p.

[6] Дейт, К. Дж. Введение в системы баз данных, 8-е издание. / К. Дж. Дейт. : пер. с англ. – М. : Издательский дом «Вильямс», 2005. – 1328 с.

[7] Роб, П. Системы баз данных : проектирование, реализация и управление. – 5-е изд., перераб. и доп. / П. Роб, К. Коронел. : пер. с англ. — СПб. : БХВ-Петербург, 2004. – 1040 с.

[8] Горшков, Д. А., Исследование современных методов проектирования базы данных / Д. А. Горшков, Л. А. Кутепова // Успехи современного естествознания. – 2011. – № 7 – С. 98-98.

[9] Типы данных столбцов [Электронный ресурс] – Режим доступа : http://phpclub.ru/mysql/doc/column-types.html.

[10] Требования к памяти для различных типов столбцов [Электронный ресурс] – Режим доступа : http://www.mysql.ru/docs/man/Storage_requirements.html.

[11] Типы данных [Электронный ресурс] – Режим доступа : http://site-do.ru/db/sql2.php.

[12] MySQL запросы : простые и сложные mysql запросы [Электронный ресурс] – Режим доступа : http://sitear.ru/material/mysql-zaprosy.

[13] Visual database creation with MySQL Workbench [Электронный ресурс] – Режим доступа : http://code.tutsplus.com/articles/visual-database-creation-with-mysql-workbench--net-10975.

[14] Using MySQL Workbench to create a database model Workbench [Электронный ресурс] – Режим доступа : http://www.techotopia.com/index.php.

[15] Методология функционального моделирования IDEF0. Руководящий документ [Электронный ресурс] – Режим доступа : http://www.nsu.ru/smk/files/idef.pdf.

[16] Диаграмма потоков данных (DFD) [Электронный ресурс] – Режим доступа : http://e-educ.ru/bd14.html.

[17] GRANT Syntax [Электронный ресурс] – Режим доступа : http://dev.mysql.com/doc/refman/5.7/en/grant.html.

[18] NodeJS Documentation [Электронный ресурс] – Режим доступа : https://nodejs.org/en/docs/

ПРИЛОЖЕНИЕ А
(обязательное)
Листинг программного кода создания схемы БД

```
CREATE DATABASE  IF NOT EXISTS `hr` /*!40100 DEFAULT CHARACTER SET utf8 */;
USE `hr`;

DROP TABLE IF EXISTS `bank_info`;
CREATE TABLE `bank_info` (
  `personnel_number` int(10) unsigned NOT NULL,
  `checking_account` varchar(45) NOT NULL,
  KEY `asdads_idx` (`personnel_number`),
  CONSTRAINT `asdads` FOREIGN KEY (`personnel_number`) REFERENCES `main_info` (`personnel_number`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
DROP TABLE IF EXISTS `candidates_info`;
CREATE TABLE `candidates_info` (
  `candidate_potential_post` smallint(5) unsigned NOT NULL,
  `candidate_firstname` varchar(45) NOT NULL,
  `candidate_lastname` varchar(45) NOT NULL,
  `candidate_middlename` varchar(45) NOT NULL,
  `candidate_age` tinyint(3) unsigned NOT NULL,
  `candidate_sex` enum('м','ж') NOT NULL DEFAULT 'м',
  `candidate_education_level` tinyint(3) unsigned NOT NULL,
  `candidate_education_degree` tinyint(3) unsigned DEFAULT NULL,
  `candidate_speciality` varchar(45) DEFAULT NULL,
  `allocation_contract_number` int(10) unsigned DEFAULT NULL,
  `candidate_phone` varchar(12) DEFAULT NULL,
  `candidate_email` varchar(100) DEFAULT NULL,
  KEY `potential_post_idx` (`candidate_potential_post`),
  KEY `education_level_idx` (`candidate_education_level`),
  KEY `education_degree_idx` (`candidate_education_degree`),
  CONSTRAINT `cadidate_education_level` FOREIGN KEY (`candidate_education_level`) REFERENCES `d_education_level` (`education_level`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `candidate_education_degree` FOREIGN KEY (`candidate_education_degree`) REFERENCES `d_education_degree` (`education_degree`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `potential_post` FOREIGN KEY (`candidate_potential_post`) REFERENCES `post_info` (`post_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
DROP TABLE IF EXISTS `contact_info`;
CREATE TABLE `contact_info` (
  `personnel_number` int(10) unsigned NOT NULL,
  `mobile` varchar(12) DEFAULT NULL,
```

```
  `phone` varchar(12) DEFAULT NULL,
  `living_address` varchar(200) DEFAULT NULL,
  `email` varchar(100) DEFAULT NULL,
  KEY `xdddd_idx` (`personnel_number`),
  CONSTRAINT `xdddd` FOREIGN KEY (`personnel_number`) REFERENCES `main_info`
(`personnel_number`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
DROP TABLE IF EXISTS `contract_info`;
CREATE TABLE `contract_info` (
  `contract_number` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `salary` bigint(20) unsigned NOT NULL,
  `contract_start_date` date NOT NULL,
  `contract_end_date` date NOT NULL,
  PRIMARY KEY (`contract_number`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
DROP TABLE IF EXISTS `d_education_degree`;
CREATE TABLE `d_education_degree` (
  `education_degree` tinyint(3) unsigned NOT NULL,
  `education_degree_descr` varchar(45) NOT NULL,
  PRIMARY KEY (`education_degree`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
DROP TABLE IF EXISTS `d_education_level`;
CREATE TABLE `d_education_level` (
  `education_level` tinyint(3) unsigned NOT NULL AUTO_INCREMENT,
  `education_level_descr` varchar(45) NOT NULL,
  PRIMARY KEY (`education_level`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;
DROP TABLE IF EXISTS `d_military_ranks`;
CREATE TABLE `d_military_ranks` (
  `military_rank` tinyint(3) unsigned NOT NULL,
  `military_rank_descr` varchar(45) NOT NULL,
  PRIMARY KEY (`military_rank`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
DROP TABLE IF EXISTS `d_relationship_status`;
CREATE TABLE `d_relationship_status` (
  `relationship_status` tinyint(3) unsigned NOT NULL,
  `relationship_status_descr` varchar(45) NOT NULL,
  PRIMARY KEY (`relationship_status`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
DROP TABLE IF EXISTS `dept_info`;
CREATE TABLE `dept_info` (
  `dept_id` smallint(5) unsigned NOT NULL AUTO_INCREMENT,
  `dept_descr` varchar(45) NOT NULL,
  `dept_head_personnel_number` int(10) unsigned DEFAULT NULL,
  PRIMARY KEY (`dept_id`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8;
```

```
DROP TABLE IF EXISTS `education_info`;
CREATE TABLE `education_info` (
  `certification_series` varchar(10) NOT NULL,
  `certification_number` int(10) unsigned NOT NULL,
  `education_level` tinyint(3) unsigned NOT NULL,
  `education_degree` tinyint(3) unsigned DEFAULT NULL,
  `qualification` varchar(45) DEFAULT NULL,
  `graduation_year` year(4) NOT NULL,
  `education_state` varchar(100) NOT NULL,
  PRIMARY KEY (`certification_series`,`certification_number`),
  KEY `education_level_idx` (`education_level`),
  KEY `education_degree_idx` (`education_degree`),
  CONSTRAINT `education_degree` FOREIGN KEY (`education_degree`) REFERENCES
`d_education_degree` (`education_degree`) ON UPDATE CASCADE,
  CONSTRAINT `education_level` FOREIGN KEY (`education_level`) REFERENCES
`d_education_level` (`education_level`) ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
DROP TABLE IF EXISTS `general_info`;
CREATE TABLE `general_info` (
  `personnel_number` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `first_name` varchar(45) NOT NULL,
  `last_name` varchar(45) NOT NULL,
  `middle_name` varchar(45) NOT NULL,
  `sex` enum('м','ж') NOT NULL DEFAULT 'м',
  `age` tinyint(3) unsigned NOT NULL,
  PRIMARY KEY (`personnel_number`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8;
DROP TABLE IF EXISTS `main_info`;
CREATE TABLE `main_info` (
  `personnel_number` int(10) unsigned DEFAULT NULL,
  `passport_id` char(14) DEFAULT NULL,
  `contract_number` int(10) unsigned DEFAULT NULL,
  `medical_card_number` int(10) unsigned DEFAULT NULL,
  `military_id` int(10) unsigned DEFAULT NULL,
  `post_id` smallint(5) unsigned DEFAULT NULL,
  `service_card_series` varchar(10) DEFAULT NULL,
  `service_card_number` int(10) unsigned DEFAULT NULL,
  `certification_series` varchar(10) DEFAULT NULL,
  `certification_number` int(10) unsigned DEFAULT NULL,
  `medical_license_number` int(10) unsigned DEFAULT NULL,
  KEY `passport_id_idx` (`passport_id`),
  KEY `contract_number_idx` (`contract_number`),
  KEY `medical_card_number_idx` (`medical_card_number`),
  KEY `military_id_idx` (`military_id`),
  KEY `post_id_idx` (`post_id`),
  KEY `service_card_series_idx` (`service_card_series`),
```

```
  KEY `service_card_number_idx` (`service_card_number`),
  KEY `certification_series_idx` (`certification_series`),
  KEY `certification_number_idx` (`certification_number`),
  KEY `medical_license_number_idx` (`medical_license_number`),
  KEY `per_number_key_idx` (`personnel_number`),
  KEY `certification_series` (`certification_series`,`certification_number`),
  KEY `service_card_series` (`service_card_series`,`service_card_number`),
  CONSTRAINT `certification_series` FOREIGN KEY (`certification_series`, `certification_number`) REFERENCES `education_info` (`certification_series`, `certification_number`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `contract_number` FOREIGN KEY (`contract_number`) REFERENCES `contract_info` (`contract_number`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `medical_card_number` FOREIGN KEY (`medical_card_number`) REFERENCES `medical_info` (`medical_card_number`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `medical_license_number` FOREIGN KEY (`medical_license_number`) REFERENCES `medical_license_info` (`medical_license_number`) ON DELETE NO ACTION ON UPDATE CASCADE,
  CONSTRAINT `military_id` FOREIGN KEY (`military_id`) REFERENCES `military_service_info` (`military_id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `passport_id` FOREIGN KEY (`passport_id`) REFERENCES `passport_info` (`passport_id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `per_number_key` FOREIGN KEY (`personnel_number`) REFERENCES `general_info` (`personnel_number`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `post_id` FOREIGN KEY (`post_id`) REFERENCES `post_info` (`post_id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `service_card_series` FOREIGN KEY (`service_card_series`, `service_card_number`) REFERENCES `service_card_info` (`service_card_series`, `service_card_number`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
DROP TABLE IF EXISTS `medical_info`;
CREATE TABLE `medical_info` (
  `medical_card_number` int(10) unsigned NOT NULL,
  `disability` tinyint(1) DEFAULT '0',
  `disability_group` tinyint(3) unsigned DEFAULT NULL,
  `disability_descr` varchar(200) DEFAULT NULL,
  `Medical_infocol` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`medical_card_number`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
DROP TABLE IF EXISTS `medical_license_info`;
CREATE TABLE `medical_license_info` (
  `medical_license_number` int(10) unsigned NOT NULL,
  `last_accreditation_date` date NOT NULL,
  `next_accreditation_date` date NOT NULL,
  PRIMARY KEY (`medical_license_number`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
DROP TABLE IF EXISTS `military_service_info`;
CREATE TABLE `military_service_info` (
  `military_id` int(10) unsigned NOT NULL DEFAULT '0',
  `military_rank` tinyint(3) unsigned DEFAULT NULL,
  `military_unit` varchar(200) DEFAULT NULL,
  `military_qualification` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`military_id`),
  KEY `military_rank_idx` (`military_rank`),
  CONSTRAINT `military_rank` FOREIGN KEY (`military_rank`) REFERENCES
`d_military_ranks` (`military_rank`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
DROP TABLE IF EXISTS `passport_info`;
CREATE TABLE `passport_info` (
  `passport_id` char(14) NOT NULL,
  `passport_number` int(7) unsigned NOT NULL,
  `passport_series` char(2) NOT NULL,
  `date_of_birth` date NOT NULL,
  `registration_address` varchar(200) NOT NULL,
  `citizenship` varchar(45) NOT NULL,
  PRIMARY KEY (`passport_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
DROP TABLE IF EXISTS `post_info`;
CREATE TABLE `post_info` (
  `post_id` smallint(5) unsigned NOT NULL AUTO_INCREMENT,
  `post_descr` varchar(45) NOT NULL,
  `required_qualification` varchar(45) DEFAULT NULL,
  `required_education` tinyint(3) unsigned NOT NULL DEFAULT '0',
  `post_dept_id` smallint(5) unsigned NOT NULL,
  `min_salary` int(10) unsigned NOT NULL,
  PRIMARY KEY (`post_id`),
  KEY `post_dept_id_idx` (`post_dept_id`),
  CONSTRAINT `post_dept_id` FOREIGN KEY (`post_dept_id`) REFERENCES `dept_info`
(`dept_id`) ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=16 DEFAULT CHARSET=utf8;
DROP TABLE IF EXISTS `post_phone_info`;
CREATE TABLE `post_phone_info` (
  `post_id` smallint(5) unsigned NOT NULL,
  `post_phone` varchar(12) NOT NULL,
  KEY `fost_id_idx` (`post_id`),
  CONSTRAINT `post_phone_id` FOREIGN KEY (`post_id`) REFERENCES `post_info`
(`post_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
DROP TABLE IF EXISTS `post_placement_info`;
CREATE TABLE `post_placement_info` (
  `post_id` smallint(5) unsigned NOT NULL,
  `post_cabinet` varchar(10) NOT NULL,
```

```
  `post_building` varchar(10) DEFAULT NULL,
  KEY `post_id_idx` (`post_id`),
  CONSTRAINT `post_placement_id` FOREIGN KEY (`post_id`) REFERENCES `post_info` (`post_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
DROP TABLE IF EXISTS `priveleges_info`;
CREATE TABLE `priveleges_info` (
  `personnel_number` int(10) unsigned NOT NULL,
  `children_quantity` tinyint(3) unsigned DEFAULT '0',
  `children_underage_quantity` tinyint(3) unsigned DEFAULT '0',
  `other_priveleges` tinyint(1) NOT NULL DEFAULT '0',
  `other_priveleges_descr` varchar(200) DEFAULT NULL,
  `relationship_status` tinyint(3) unsigned DEFAULT '0',
  KEY `personnel_number_idx` (`personnel_number`),
  KEY `relationship_status_idx` (`relationship_status`),
  CONSTRAINT `priveleges_personnel_number` FOREIGN KEY (`personnel_number`) REFERENCES `main_info` (`personnel_number`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `relationship_status` FOREIGN KEY (`relationship_status`) REFERENCES `d_relationship_status` (`relationship_status`) ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
DROP TABLE IF EXISTS `service_card_info`;
CREATE TABLE `service_card_info` (
  `service_card_series` varchar(10) NOT NULL,
  `service_card_number` int(10) unsigned NOT NULL,
  `service_card_date_of_issue` date NOT NULL,
  `service_card_profession` varchar(45) DEFAULT NULL,
  `service_card_education` varchar(45) NOT NULL,
  `service_card_speciality` varchar(45) DEFAULT NULL,
  `service_card_liner_series` varchar(10) DEFAULT NULL,
  `service_card_liner_number` int(10) unsigned DEFAULT NULL,
  `service_card_liner_date_of_issue` date DEFAULT NULL,
  PRIMARY KEY (`service_card_series`,`service_card_number`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
DELIMITER ;;
CREATE DEFINER=`root`@`localhost` PROCEDURE `cl_createUser`(
        in _last_name varchar(45),
  in _first_name varchar(45),
  in _middle_name varchar(45),
  in _sex enum('м','ж'),
  in _age tinyint,
  in _post_id smallint unsigned,
  in _medical_license_number int unsigned,
  in _last_accreditation_date date,
  in _next_accreditation_date date)
begin
        declare _personnel_number int unsigned;
```

```
            insert into general_info
                (last_name, first_name, middle_name, sex, age)
            values
                (_last_name, _first_name, _middle_name, _sex, _age);

            set _personnel_number = last_insert_id();

            if (_medical_license_number is null) then
                insert into main_info
                    (personnel_number, post_id)
                values
                    (_personnel_number, _post_id);
            end if;
        if (_medical_license_number is not null) then
                insert into medical_license_info
                    (medical_license_number, last_accreditation_date,
next_accreditation_date)
                values
                    (_medical_license_number, _last_accreditation_date,
_next_accreditation_date);

                insert into main_info
                    (personnel_number, post_id, medical_license_number)
                values
                    (_personnel_number, _post_id, _medical_license_number);
            end if;
    end ;;
    DELIMITER ;
    DELIMITER ;;
    CREATE DEFINER=`root`@`localhost` PROCEDURE `cl_deleteUser`(in
_personnel_number int unsigned)
    begin
      declare _medical_license_number int;
      select medical_license_number into _medical_license_number from main_info where
personnel_number=_personnel_number;

            delete from main_info where personnel_number = _personnel_number;
            delete from general_info where personnel_number = _personnel_number;
        delete from medical_license_info where medical_license_number =
_medical_license_number;
    end ;;
    DELIMITER ;
    DELIMITER ;;
    CREATE DEFINER=`root`@`localhost` PROCEDURE `cl_getLicenseInfo`()
    begin
```

```
		select
		general.last_name,
		general.first_name,
		general.middle_name,
		main.medical_license_number,
		date_format(medical_license_info.last_accreditation_date, '%d-%m-%Y') as
last_accreditation_date,
		date_format(medical_license_info.next_accreditation_date, '%d-%m-%Y') as
next_accreditation_date
			from main_info as main
				inner join general_info as general
					on general.personnel_number = main.personnel_number
				inner join medical_license_info
					on main.medical_license_number =
medical_license_info.medical_license_number;
	end ;;
	DELIMITER ;
	DELIMITER ;;
	CREATE DEFINER=`root`@`localhost` PROCEDURE `cl_getPostInfo`()
	begin
		select general_info.last_name, general_info.first_name,
general_info.middle_name, post_info.post_descr
		from main_info
			inner join general_info
				on main_info.personnel_number = general_info.personnel_number
			inner join post_info
				on main_info.post_id = post_info.post_id;
	end ;;
	DELIMITER ;
	DELIMITER ;;
	CREATE DEFINER=`root`@`localhost` PROCEDURE `cl_getStuffInfo`()
	begin
		select * from general_info;
	end ;;
	DELIMITER ;
	DELIMITER ;;
	CREATE DEFINER=`root`@`localhost` PROCEDURE `cl_updateUser`(in
_personnel_number int unsigned,
	  in _last_name varchar(45),
	  in _first_name varchar(45),
	  in _middle_name varchar(45),
	  in _sex enum('м','ж'),
	  in _age tinyint,
	  in _post_id smallint unsigned,
	  in _medical_license_number int unsigned,
	  in _last_accreditation_date date,
```

```
    in _next_accreditation_date date)
  begin
    declare _medical_license_number int;
    select medical_license_number into _medical_license_number from main_info where personnel_number=_personnel_number;

    if (_last_name is not null) then update general_info set last_name = _last_name where personnel_number = _personnel_number;
    end if;

    if (_first_name is not null) then update general_info set first_name = _first_name where personnel_number = _personnel_number;
    end if;

    if (_middle_name is not null) then update general_info set middle_name = _middle_name where personnel_number = _personnel_number;
    end if;

    if (_age is not null) then update general_info set age = _age where personnel_number = _personnel_number;
    end if;

    if (_sex is not null) then update general_info set sex = _sex where personnel_number = _personnel_number;
    end if;

    if (_post_id is not null) then update main_info set post_id = _post_id where personnel_number = _personnel_number;
    end if;

    if (_medical_license_number is not null) then
        update medical_license_info set medical_license_number = _medical_license_number
            where medical_license_number in
            (select medical_license_number from  main_info where personnel_number = _personnel_number);
    end if;

    if (_last_accreditation_date is not null) then
        update medical_license_info set last_accreditation_date = _last_accreditation_date
            where medical_license_number in
            (select medical_license_number from  main_info where personnel_number = _personnel_number);
    end if;

    if (_next_accreditation_date is not null) then
```

```
            update medical_license_info set next_accreditation_date =
_next_accreditation_date
                    where medical_license_number in
                    (select medical_license_number from  main_info where personnel_number
= _personnel_number);
        end if;

    end ;;
    DELIMITER ;
    DELIMITER ;;
    CREATE DEFINER=`root`@`localhost` PROCEDURE `CreateUser`(
            in _last_name varchar(45),
        in _first_name varchar(45),
        in _middle_name varchar(45),
        in _sex enum('м','ж'),
        in _age tinyint,
        in _salary bigint unsigned,
        in _contract_start_date date,
        in _contract_end_date date,
            in _passport_id char(14),
        in _passport_number int(7) unsigned,
        in _passport_series char(2),
        in _date_of_birth date,
        in _registration_address varchar(200),
        in _citizenship varchar(45),
        in _medical_license_number int unsigned,
        in _last_accreditation_date date,
        in _next_accreditation_date date)
    begin
            declare _personnel_number int unsigned;
            declare _contract_number int unsigned;

            insert into general_info
                    (last_name, first_name, middle_name, sex, age)
            values
                    (_last_name, _first_name, _middle_name, _sex, _age);

            set _personnel_number = last_insert_id();

            insert into contract_info
                    (salary, contract_start_date, contract_end_date)
            values
                    (_salary, _contract_start_date, _contract_end_date);

            set _contract_number = last_insert_id();
```

```
		insert into passport_info
			(passport_id, passport_series, passport_number, date_of_birth,
registration_address, citizenship)
		values
			(_passport_id, _passport_series, _passport_number, _date_of_birth,
_registration_address, _citizenship);

		if (_medical_license_number is null) then
			insert into main_info
				(personnel_number, contract_number, passport_id)
			values
				(_personnel_number, _contract_number, passport_id);
		end if;
	    if (_medical_license_number is not null) then
			insert into medical_license_info
				(medical_license_number, last_accreditation_date,
next_accreditation_date)
			values
				(_medical_license_number, _last_accreditation_date,
_next_accreditation_date);

			insert into main_info
				(personnel_number, contract_number, passport_id,
medical_license_number)
			values
				(_personnel_number, _contract_number, passport_id,
_medical_license_number);
		end if;
	end ;;
	DELIMITER ;
	DELIMITER ;;
	CREATE DEFINER=`root`@`localhost` PROCEDURE `DeleteUser`(in _personnel_number
int unsigned)
	begin
		delete from general_info where personnel_number = _personnel_number;

	    delete from contract_info where contract_number in
			(select contract_number from main_info where personnel_number =
_personnel_number);

		delete from medical_license_info where medical_license_number in
			(select medical_license_number from main_info where personnel_number
= _personnel_number);

		delete from passport_info where passport_id in
```

```
                    (select passport_id from main_info where personnel_number =
_personnel_number);


                delete from bank_info where personnel_number = _personnel_number;

                delete from contact_info where personnel_number = _personnel_number;

                delete from priveleges_info where personnel_number = _personnel_number;

                delete from education_info
                        where certification_number in
                                (select certification_number from main_info where
personnel_number = _personnel_number)
                        and certification_series in
                                (select certification_series from main_info where personnel_number
= _personnel_number);

                delete from service_card_info
                        where service_card_number in
                                (select service_card_number from main_info where
personnel_number = _personnel_number)
                        and service_card_series in
                                (select service_card_series from main_info where
personnel_number = _personnel_number);

                delete from medical_info where medical_card_number in
                        (select medical_card_number from main_info where personnel_number =
_personnel_number);

                delete from military_service_info where military_id in
                        (select military_id from main_info where personnel_number =
_personnel_number);

                -- final
                delete from main_info where personnel_number = _personnel_number;
        end ;;
        DELIMITER ;
```

ПРИЛОЖЕНИЕ Б
(обязательное)
Листинги некоторых методов работы клиентского приложения

Методы для формирования HTTP-запроса на сервер:

```
angular.module('app', ['pascalprecht.translate', 'app-i18n']).
	service('dbService', ['$http', '$rootScope', function ($http, $rootScope){
		var performGetQuery = function (query){
			return $http({
				method : 'GET',
				url : '/db',
				headers : {
					"query" : query
				}
			})
			.then(function (response){
				console.log("from db : ", response);
				var fields = response.data.fields.map(function (field){
					return field.name;
				})
				return {
					fields : fields,
					rows : response.data.rows
				}
			}, function (error){
				console.log ("impossible to perform query : ", error);
				alert('error');
				return {
					fields : null,
					rows : null
				}
			})
		}
		var performGetProcedure = function (query){
			return $http({
				method : 'GET',
				url : '/db',
				headers : {
					"query" : query
				}
			})
			.then(function (response){
				console.log("from db : ", response);
				var fields = response.data.fields[0].map(function (field){
					return field.name;
```

```
                        })
                        return {
                                fields : fields,
                                rows : response.data.rows[0]
                        }
                }, function (error){
                        console.log ("impossible to perform query : ", error);
                        return {
                                fields : null,
                                rows : null
                        }
                })
        }
        var performSetProcedure = function (query){
                return $http({
                        method : 'POST',
                        url : '/db',
                        headers:    {'Content-Type':    'application/x-www-form-
urlencoded'},
                        data: { "query" : query }
                })
                .then(function (response){
                        console.log("from db : ", response);
                        $rootScope.$broadcast('refresh');
                }, function (error){
                        console.log ("impossible to perform query : ", error);
                        $rootScope.$broadcast('refresh');
                })
        }
        return {
                performGetQuery : performGetQuery,
                performGetProcedure : performGetProcedure,
                performSetProcedure : performSetProcedure
        }
}])
```

ПРИЛОЖЕНИЕ В
(обязательное)
Листинги некоторых методов работы серверного приложения

Авторизация пользователей и обращение к базе для выполнения запроса:

```
var bodyParser = require('body-parser');
var mysql = require("mysql");
var express = require('express');
var app = express();
var server;

app.set('view engine', 'html');
app.use(function (req, res, next){
	console.log("%s %s", req.method, req.url);
	next();
});
app.use("/node_modules", express.static('node_modules'));
app.use(express.static(__dirname + "/public", { redirect : false }));
app.use(bodyParser.urlencoded({extended: true}));


var connection;
app.get('/auth', function (req, res){
	if (req.headers.logout){
		connection.end(function (err){
			res.end('server : logout')
		})
	}
	var username = req.headers.username;
	var password = req.headers.password;
	console.log("login attempt: ")
	console.log("user:", username);
	console.log("pwd:", password);
	connection = mysql.createConnection({
		host : "localhost",
		user : username,
		password : password,
		database  : "hr"
	})
	connection.connect(function (err){
		if (err){
			console.log('access denied');
			res.end(JSON.stringify({
				allowed : false
```

```
                        }));
                        return;
                }
                console.log('successfully logged in');
                res.end(JSON.stringify({
                        allowed : true
                }));
        })
})

app.route('/db')
        .get(function (req, res){
                var query = req.headers.query;
                connection.query(query, function (err, rows, fields){
                        console.log('performing query...', query);
                        res.json({
                                rows : rows,
                                fields : fields
                        })
                })
        }).
        post(function (req, res){
                var query = JSON.parse(Object.keys(req.body)[0]).query; //ta-dam!)
                connection.query(query, function (err, rows, fields){
                        console.log('performing query...', query);
                        res.end();
                })
                console.log(JSON.parse(query).query)
        })

app.get('/:page?', function (req, res) {
        res.redirect('/');
        res.sendFile(__dirname + '/public/index.html');
})


server = app.listen(3000, function (){
        console.log("Server started successfully");
});
```

ПРИЛОЖЕНИЕ Г
(обязательное)
Графическая часть