МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий

Кафедра информационно-измерительных систем

Направление подготовки: 230100 Информатика и вычислительная техника

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ БАКАЛАВРСКАЯ РАБОТА

Лапшин Александр Дмитриевич

Разработка клиентского модуля программно-аппаратного комплекса хранения и обработки биомедицинских данных

«К защите допущен»
Потатуркин О. И.
д. техн. наук, профессор

_____

«……»……2014г.

Хазанкин Г. Р,
системный инженер НИИ ФФМ СО РАМН
б/с

_____

«……»……2014г.

Дата защиты: «……»……2014г.

Автор: Лапшин А.Д. _____

Новосибирск, 2014г.

Оглавление

Введение 3
Глава 1 4
§1.1 Классификация и средства проектируемой информационной системы 4
§1.2 Обзор существующих решений 5
Глава 2 8
§2.1 Специфика 8
§2.2 Система управления базами данных (СУБД) 9
2.2.1 Требования. 9
2.2.2 MongoDB - документо-ориентированная СУБД. 10
§2.3 Формат представления данных в БД 11
2.3.1 Основные документы: “Испытуемый”, “Исследователь”, “Исследование”. 12
2.3.2 Побочные объекты: “Секция”, “Задача”. 14
2.3.3 Разграничение прав доступа. 17
2.3.4 Дополнительно: Словарь. 19
2.3.5 Листинги основных объектов. 20
Глава 3 24
§3.1 Добавление, удаление, изменение и чтение данных в системе. 24
3.1.1 Добавление. 24
Секции. 25
Исследователи. Роли. 26
Исследования. Задачи. Испытуемые. 28
Словари. 28
3.1.2 Изменение и удаление. 29
Удаление. 29
Изменение. 30
3.1.3 Чтение. 34
§3.2 Визуализации процесса распределения задач связанных с обработкой психофизиологических данных. 38
Заключение 41
Список источников 42

Введение.

Перед учеными Научно исследовательского института физиологии и фундаментальной медицины Сибирского отделения Российской академии медицинских наук (НИИ ФФМ СО РАМН) встала непростая задача автоматизации процессов управления научными исследованиями.

Причиной этому послужило стремительное увеличение объемов обрабатываемых данных, а также географическая распределенность лабораторий и институтов совместно с которыми проводятся исследования (ученые проводят исследования не только на территории самого института, но и совместно с коллегами, находящимися в различных странах мира).

В связи с этим принято решение разработать программно-аппаратный комплекс для хранения и обработки биомедицинских данных. Задачей комплекса является не только безопасное хранение, но и предоставление инструментов для удобной выборки и добавления исследовательских данных, а также обеспечение быстрой передачи таких данных, между исследователями.

Целью данной дипломной работы, является разработка проекта клиентского модуля данного программно-аппаратного комплекса. Поставлены следующие задачи:

- проанализировать специфику
- сформулировать требования к системе управления базами данных (СУБД)
- определить СУБД, удовлетворяющую данным требованиям
- определить формат представления данных в БД
- спроектировать клиентское приложение, позволяющее ученым автоматизировать и упростить управление исследованиями (интерфейс оператора).

Важной спецификой работы, является разнородность исследовательских данных (радиологические, электрофизиологические, эндоскопические, психометрические и др.). Человек может участвовать в различных исследованиях и, таким образом, набор данных для каждого испытуемого будет индивидуальным. В связи с этим, система должна быть достаточно гибкой и работать с произвольными наборами данных, связанных с испытуемыми. Также система должна позволять пользователю добавлять новые типы исследовательских данных, без вмешательства разработчика.

Глава 1

§1.1 Классификация и средства проектируемой информационной системы

Международный стандарт ИСО/МЭК 2382-1:1993 определяет информационную систему (ИС) как систему обработки информации, работающую совместно с организационными ресурсами, такими как люди, технические средства и финансовые ресурсы, которые обеспечивают и распределяют информацию [3].

Выделяют следующие виды классификаций информационный систем:

- Классификация по архитектуре
- Классификация по степени автоматизации
- Классификация по характеру обработки данных
- Классификация по сфере применения
- Классификация по охвату задач (масштабности)

Разработанную ИС можно классифицировать следующим образом:

- По архитектуре. Разработанную ИС можно классифицировать как: ***распределенная***, ***клиент-серверная***, ***многозвенная***. Данная ИС выполнена в виде веб-приложения. В таких приложениях помимо звена СУБД и клиентского звена, выполняющегося в веб-браузере, имеется промежуточное звено — веб-сервер с соответствующим серверным программным обеспечением.
- По степени автоматизации. Разработанную ИС можно классифицировать как ***автоматизированную***, так как она призвана автоматизировать процесс постановки исследовательских задач, связанных с обработкой радиологических, электрофизиологических, эндоскопических, психометрических и др. данных испытуемых. Такая ИС требует участия человека.
- По характеру обработки данных. Разработанную ИС, можно классифицировать как ***информационно-поисковую***. В данной ИС не используются сложные алгоритмы для обработки данных. Основной целью является правильная выдача данных испытуемого исследователю в соответствие с его правами доступа, и обеспечение различных выборок этих данных.

- По сфере применения. Разработанную ИС, можно классифицировать как ***научно-исследовательскую медицинскую*** ИС. Основной сферой использования разработанной ИС, является научно-исследовательская сфера. Так как исследования проводятся на базе НИИ ФФМ, она подходит под приведенную классификацию.
- По охвату задач. Разработанную ИС, можно классифицировать как ***групповую***. Каждое исследование, как правило проводится определенной группой исследователей, из которых каждый может находится обособленно от других и в другой стране.

В разработанной ИС предусмотрены:

- средства хранения и передачи медицинских данных
- средства управления научными исследованиями и интерфейс взаимодействия пользователя с системой

§1.2 Обзор существующих решений

Так как разработанную ИС можно частично классифицировать как медицинскую информационную систему (МИС), стоит лучше рассмотреть данный вид систем. МИС - комплексная автоматизированная информационная система для автоматизации деятельности ЛПУ, в которой объединены система поддержки принятия медицинских решений, электронные медицинские записи о пациентах, данные медицинских обследований в цифровой форме, данные мониторинга состояния пациента с медицинских приборов, средства общения между сотрудниками, финансовая и административная информация.

В соответствии со стандартом организации СТО МОСЗ 91500.16.0002-2004 «Информационные системы в здравоохранении. Общие требования» [4], МИС можно классифицировать следующим образом:

1. Медико-технологические ИС (МТИС), предназначенные для информационного обеспечения процессов диагностики, лечения, реабилитации и профилактики пациентов в лечебно-профилактических учреждениях.
2. Информационно-справочные системы (БИИС), содержащие банки медицинской информации для информационного обслуживания медицинских учреждений и служб управления здравоохранением.

3. Статистические ИС (СМИС) органов управления здравоохранением.
4. Научно-исследовательские ИС (НИИС), предназначенные для информационного обеспечения медицинских исследований в клинических научно-исследовательских институтах (НИИ).
5. Обучающие ИС (ОМИС), предназначенные для информационного обеспечения процессов обучения в медицинских учебных заведениях.

Пользователями ИС указанных классов являются учреждения и организации системы здравоохранения, в соответствии с [4]:

- больницы (участковые сельские, районные, городские, областные, ведомственные);
- клинические НИИ;
- поликлиники (районные, городские, ведомственные);
- территориальные специализированные медицинские службы;
- санаторно-курортные учреждения;
- органы управления здравоохранением на федеральном (Минздрав России) и территориальном уровнях (ОТУЗ), ведомственные медицинские управления других министерств и ведомств (МУВ);
- территориальные органы Госсанэпиднадзора;
- организации фондов обязательного медицинского страхования на федеральном и территориальном уровнях, страховые медицинские организации;
- медицинские учебные заведения.

Cerner Corporation

Рассмотрим МИС предоставляемую компанией CernerCorporation. В рамках данной МИС пациенту выдается магнитная карта. Карта предосталяется не самой корпорацией Cerner, а так называемым поручителем. В качестве поручетеля может выступать, как страховая компания, так и работодатель пациента.

Все медицинские данные, собранные пациентом с момента получения карты, хранятся в удаленной БД на серверах CernerCorporation, в зашифрованном виде. Магнитная карта, является своего рода ключом, для доступа к данным пациента.

Когда пациент обращается в какое либо учреждение, в котором могут потребоваться его медицинские данные: ЛПУ, аптека или оздоровительный центр - карта проводится через

считывающее устройство и происходит запрос к серверу. Совместно с идентификатором пользователя, на сервер также отправляются данные учреждения, запросившего данные, чтобы проверить, имеет ли право данное учреждение, получить доступ к персональным данным пациента.

После авторизации учреждение получает доступ к персональным данным пациента и человек может, к примеру, с легкостью обратиться к врачу, все данные уже будут загружены на его компьютер. После обращения к врачу и назначения каких либо лекарств, пациент может обратиться в аптеку, и если аптека также работает с данными картами, он также с легкостью и быстро получит нужные медикаменты.

Основные подходы МИС от CernerCorporation:

- спонсорская система выдачи карт пациенту
- централизованное хранение персональных (медицинских) данных пациента
- быстрый и защищенный доступ к этим данным

Также МИС от CernerCorporation имеет научно-исследовательский модуль ClinicalResearch. Данный модуль позволяет производить различные выборки по медицинским данным пациентов. К примеру, можно получить данные всех пациентов в возрасте от 30 до 50 лет, болеющих диабетом. При этом эти данные будут предоставлены в анонимезированном виде, тем самым личность пациента остается закрытой для исследователя. Но к сожалению, данная МИС не предоставляет функционала управления научными исследованиями, а лишь предоставляет возможность выборки. Именно по этой причине данная система не соответствует поставленным требованиям.

Вывод

МИС изначально ориентированы на сбор и хранения данных в карте пациента и лишь немногие из них, такие как МИС от CernerCorporation, предоставляют возмоность произведения выборок анонимезированных данных.

На данный момент все МИС, включая рассмотренную, являются проприетарными и в них не предусмотрен функционал системы управления науными исследованиями.

Глава 2

Данная глава состоит из трех основных параграфа с названиями: *Специфика*; *Система Управления Базами Данных*; *Формат представления данных в БД*.

В параграфе *Специфика* будет рассмотрена специфика данной задачи, для наилучшего понимания будет приведен пример.

В параграфе *Система Управления Базами Данных* сформулированы, основанные на специфике, требования к СУБД. После чего приведен краткий обзор СУБД, определенной в соответствие с данными требованиями.

В параграфе *Формат представления данных в БД*, приведено подробное описания предложенного формата представления данных в БД, с подробным обоснованием основных проблем и принятых, на их основе, решений.

§2.1 Специфика

Спецификой проекта является хранение и обработка больших объемов разнородных данных. Рассмотрим пример хранения электронного досье двух испытуемых (*“Испытуемый 1”*, *“Испытуемый 2”*). У каждого имеется набор следующих полей: ФИО; Дата рождения; Регион проживания; - этот набор являются базовым для каждого испытуемого; эти поля должны быть заполнены у всех, кроме анонимных случаев. Но об испытуемом необходимо хранить не только базовые данные, но и множество других. Например, болезни которыми в детстве болел *“Испытуемый 1”* или каким видом спорта занимается / занимался *“Испытуемый 2”*. Причем набор данных может быть различным. У каждого исследования набор этих данных может увеличиваться и этот набор для каждого испытуемого будет индивидуальным. Именно этот набор данных (индивидуальный для каждого испытуемого), собранный из разных исследований и является *метаданными* испытуемого.

Также особенностью задачи является необходимость визуализации процесса распределения задач связанных с обработкой психофизиологических данных: т.е. требуется построить определенную схему, используя которую старший научный сотрудник, возглавляющий проект, сможет легко увидеть кому и кем даны определенные задания и следить за ходом их выполнения.

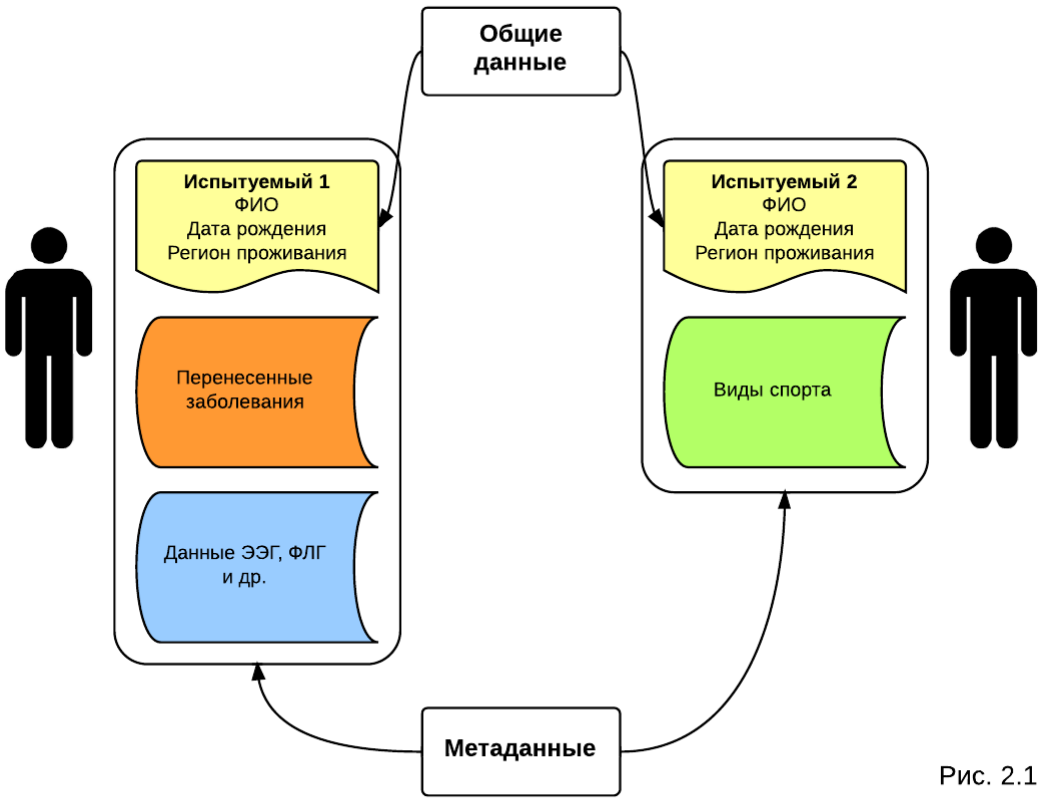

Рис. 2.1

§2.2 Система управления базами данных (СУБД)

2.2.1 Требования.

Помимо основных общепринятых требований к БД следует, основываясь на приведенной выше специфике задачи, вывести собственные дополнительные требования, которым должна удовлетворять БД и соответственно СУБД управляющая ей. Но сначала, для удобства описания данных требований, следует ввести некоторые определенные термины:

- *Карточка испытуемого* - набор метаданных испытуемого, включая сопутствующие файлы с результатами обследований и заключениями исследователей

Требуемые задачей свойства БД и соответственно ее СУБД приведены в виде операций, которые СУБД должна уметь совершать над данными, хранящимися в БД:

- Карточка испытуемого может иметь различное количество полей. Данное требование основано на примере, приведенном на **Рис.2.1**. Каждый испытуемый имеет различный набор метаданных.
- Должна присутствовать возможность добавления новых “пользовательских”

полей в карточку испытуемого. То есть должна присутствовать возможность простого расширения набора метаданных испытуемого.

2.2.2 MongoDB - документо-ориентированная СУБД.

Основываясь на приведенных выше требованиях, в качестве СУБД решено выбрать MongoDB. MongoDB - это кроссплатформенная документо-ориентированная СУБД с открытым кодом.

Выбор именно документо-ориентированной СУБД основан на том, что в отличие от реляционных аналогов здесь не накладываются ограничения на набор полей у документа. Так как документо-ориентированные базы данных еще не так распространены как реляционные, но их популярность с каждым годом все увеличивается, целесообразно будет разъяснить некоторые нюансы терминологии документо-ориентированных баз данных. В кратком учебнике по MongoDBTheLittleMongoDBBook от автора KarlSeguin[6]приведены шесть основных концепций MongoDB:

1. MongoDB — концептуально то же самое, что обычная, привычная нам база данных (или в терминологии Oracle — схема). Внутри MongoDB может быть ноль или более баз данных, каждая из которых является контейнером для прочих сущностей.
2. База данных может иметь ноль или более «коллекций». Коллекция является аналогом таблицы из реляционных БД. Коллекции состоят из нуля или более «документов». Документ является аналогом записи в таблице (строки). Документ состоит из одного или более «полей», поля, в свою очередь, являются аналогом “колонки”.
3. «Индексы» в MongoDB почти идентичны таковым в реляционных базах данных.
4. «Курсоры» отличаются от предыдущих пяти концепций, но они очень важны (хотя порой их обходят вниманием) и заслуживают отдельного обсуждения. Важно понимать, что когда мы запрашиваем у MongoDB какие-либо данные, то она возвращает курсор, с которыми мы можем делать все что угодно — подсчитывать, пропускать определённое число предшествующих записей — при этом не загружая сами данные.

Основное же отличие документо-ориентированных баз данных от реляционных

является то, что в реляционных базах данных «колонки» определяются на уровне «таблицы», тогда как документо-ориентированных базы данных определяют «поля» на уровне «документа». То есть любой документ внутри коллекции может иметь свой собственный уникальный набор полей, что и является ключевым аспектом, делающим документо-ориентированные базы данных наиболее подходящими для данной задачи.

Выбор пал именно на MongoDB по причине ее развитости, популярности и наличия хорошей и полной документации. Основными плюсами MongoDB для данного проекта, помимо ее документо-ориентированности, стали:

- схожий с реляционными СУБД подход к хранению данных: базаданных-коллекция-документ-поле. К примеру, у основного конкурента MongoDB, CouchDB, отсутствует понятие какого-либо разделения документов на группы и все документы являются равнозначными, а для разделения документов, например, по типу, необходимо добавлять специальные поля.
- формат хранения данных. Данные в MongoDB хранятся в формате BSON. BSON - это практически тоже самое, что и JSON (распрастраненный формат представления данных в web), но в бинарном виде. Тот факт, что данные хранятся именно в бинарном виде, значительно ускоряет время их обработки.
- существует инструмент GridFS, позволяющий хранить в БД файлы любого размера. Стоит отметить, что на данном этапе разработки, файлы больших размеров, такие, например, как DICOM-файлы, предлагается хранить в другой, более защищенной и предназначенной для этого БД, под управлением специализированной СУБД, например Oracle. Но, несомненно, наличие такого инструмента является огромным плюсом.
- надо еще???

§2.3 Формат представления данных в БД

Так как в качестве СУБД была выбрана документо-ориентированная MongoDB, то и структура базы данных будет приводится в ее терминологии. Описание, для большей наглядности и понимания, приведено в порядке развертывания от основных документов к побочным. Далее рассмотрены каким образом решена задача разграничения прав доступа. Также приведено частичное решение проблемы локализации основных элементов данного формата. В конце параграфа приведены листинги с моделями основных документов с

комментариями.

2.3.1 Основные документы: “Испытуемый”, “Исследователь”, “Исследование”.

Основными документами в базе данных являются: *“Испытуемый”* (“Testsubject”), *“Исследователь”* (“Researcher”) и *“Исследование”* (“Research”). На **Рис.2.2** наглядно показаны связи между тремя основными объектами базы данных. Все документы хранятся в своей отдельной коллекции, соответственно “Испытуемые”, “Исследователи”, “Исследования”.

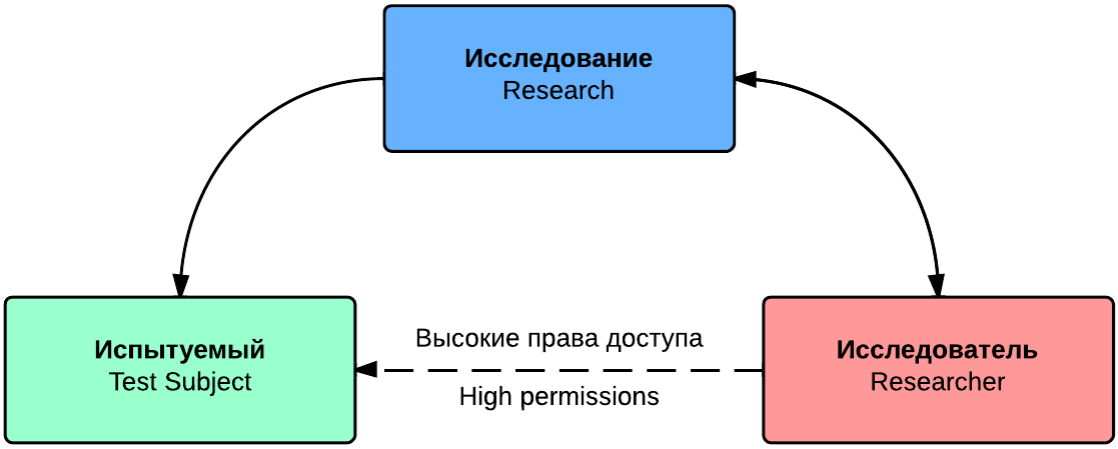

Рис. 2.2

Исследователь

Документ *“Исследователь”* является аккаунтом реального научного сотрудника и содержит в себе:

- уникальный идентификатор
- общие данные самого научного сотрудника: ФИО, возраст, пол, лаборатория в которой он работает и т.д.
- оборудование и методы обследований, которые сотрудник может использовать
- текущий статус занятости: занят, свободен и т.д.
- роль исследователя, регламентирующая права доступа, которыми он обладает
- флаги базы данных: удален/не удален, дата создания документа (аккаунта), дата последнего изменения и т.д. Подробнее в §3.1

Испытуемый

Документ *“Испытуемый”* не имеет никаких прав доступа и является отображением карточки испытуемого, то есть лишь хранит набор метаданных каждого реального испытуемого для использования их в ходе различных исследований. Не представляется возможным описать полнуюструктуру данных данного объекта, так как он может содержать в себе различный набор секций (*“Секция”* см. раздел 2.3.2 *Секция*), в которых хранятся различные данные связанные с ним. Можно лишь выделить набор секций который будет обязателен для каждого экземпляра документа *“Испытуемого”*:

- уникальный идентификатор
- общие данные испытуемого: ФИО, пол, возраст. При анонимном опрашивание ФИО отсутствует
- флаги базы данных: удален/не удален, дата создания документа, дата последнего изменения и т.д.

Исследование

Документ *“Исследование”* является связующим звеном, предоставляющим *“Исследователям”* доступ к данным определенного набора *“Испытуемых”*. Только через него большинство научных сотрудников (*“Исследователей”* в БД) смогут получить доступ к данным *“Испытуемого”*, исключение могут составлять только *“Исследователи”* имеющие приоритетные права доступа к коллекции *“Испытуемых”*. Документ *“Исследование”* содержит в себе:

- уникальный идентификатор
- общую информацию об исследование: название, цель, инициатор(лаборатория или сотрудник)
- список *“Исследователей”* работающих над исследованием
- список *“Испытуемых”* участвующих в исследовании
- флаги базы данных

2.3.2 Побочные объекты: “Секция”, “Задача”.

Секция

Для удобства структурирования данных хранящихся внутри основных объектов, используются секции. Структура документов в MongoDB позволяет хранить документы внутри документов, такой подход в MongoDB называется денормализованная модель данных.

“Секция” является отдельным документом и хранится внутри основных документов (*“Испытуемый”*, *“Исследователь”*, *“Испытание”*) или же проще говоря, основные документы состоят из *“Секций”*. На **Рис.2.3** схематично показана структура основных документов разделенных на секции, сами секции изображены желтым цветом.

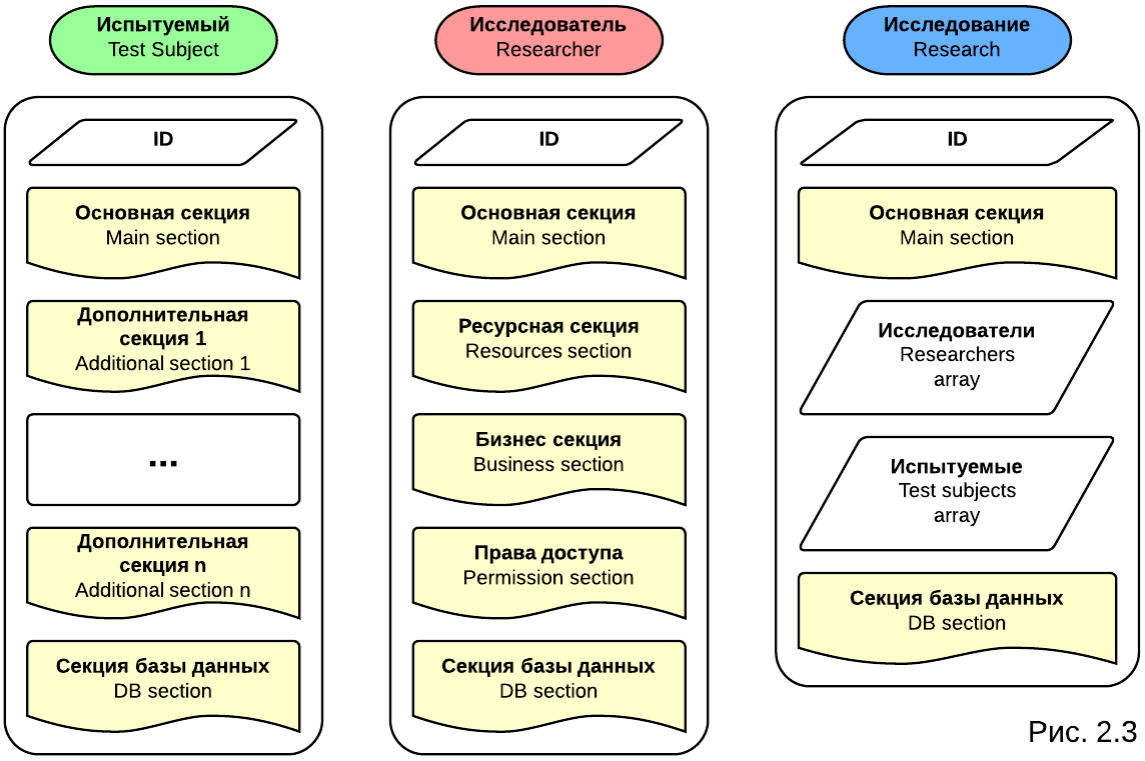

Рис. 2.3

Важно отметить тот факт, что основные документы не ссылаются на свои *“Секции”* , а каждая *“Секция”* хранится именно внутри своего основного документа. Также стоит заметить, что для хранения списка *“Исследователей”* и *“Испытуемых”* в *“Исследование”* используются массивы (данный тип данных также присутствует в MongoDB), в которых уже как раз будут хранится ссылки на соответствующие объекты.

Как можно заметить на **Рис.2.3** документ *“Испытуемый”* имеет различный набор секции. Дополнительные секции создаются пользователем (научным сотрудником) у

которого есть права для добавления новых и изменения данных уже существующих *“Испытуемых”*.

Для того чтобы сохранять созданные пользователем новые типы секций и исключить создание дубликатов, создается коллекция секций. Точнее говоря коллекция типов секций, в которой хранятся “скелеты” уже созданных секций. В **Листинге 2.1** приведена структура документа *“Секция”* в формате JSON (в подобном виде она хранится в MongoDB), в комментариях приведены пояснения основных полей.

```
        {
_id : <Section-type-ID>, // идентификатор типа секции, задается явно при добавлении
sec_data : {             // данные секции также хранятся как отдельный документ внутри нее
data1 : <Some-Data>,
data2 : <Some-Data>, // данные могут иметь любой доступный в MongoDB тип, включая массивы
data3 : <Some-Data>,
        ...
dataN : <Some-Data>
    }
}
```

Листинг 2.1. Модель документа *“Секция”*

Задача

“Задачи” (“Issues”) являются необходимым документом для распределения заданий между исследователями. Каждый исследователь может назначить определенную *“задачу”* другому исследователю, если только этот исследователь не выше его в правах доступа. *“Задача”* также может быть назначена и самому себе, к примеру, нужно чтобы определенная деятельность была зафиксирована и ее, на данный момент, может выполнить только сам исследователь, тогда он просто назначает сам себе *“задачу”*.

Все задачи хранятся в отдельной коллекции. В теле задачи хранится ссылка на документ исследователя, которому она назначена, но в теле документа *“Исследователь”* не хранятся ссылки на задачи, которые назначены ему. Данный подход решает проблему образования ложных связей между данными документами и избежать возможных ошибок в будущем. Для наилучшего понимания преимущества такого метода, и что такое ложные связи, рассмотрим пример:

На **Рис.2.4** приведены примеры двух подходов к хранению связи между *“Исследователями”* и *“Задачами”*. На **Рис.2.4.а**синим, обозначены ссылки хранящиеся в документе *“Задача”*, ведущие к документу *“Исследователь”*, а черным, обратные ссылки, хранящиеся в документе *“Исследователь”*. Таким образом каждый документ

“Исследователь” знает список всех задач, назначенных ему, а каждый документ “Задача” знает кому она назначена. Но что будет, если задача будет переназначена другому исследователю, то есть ссылка будет изменена на документ другого *“Исследователя”*. Тогда произойдет ситуация, показанная на **Рис.2.4.b**, в котором красным показана ссылка из *“Исследователя”*, на *“Задачу”*, которая ему больше не назначена.

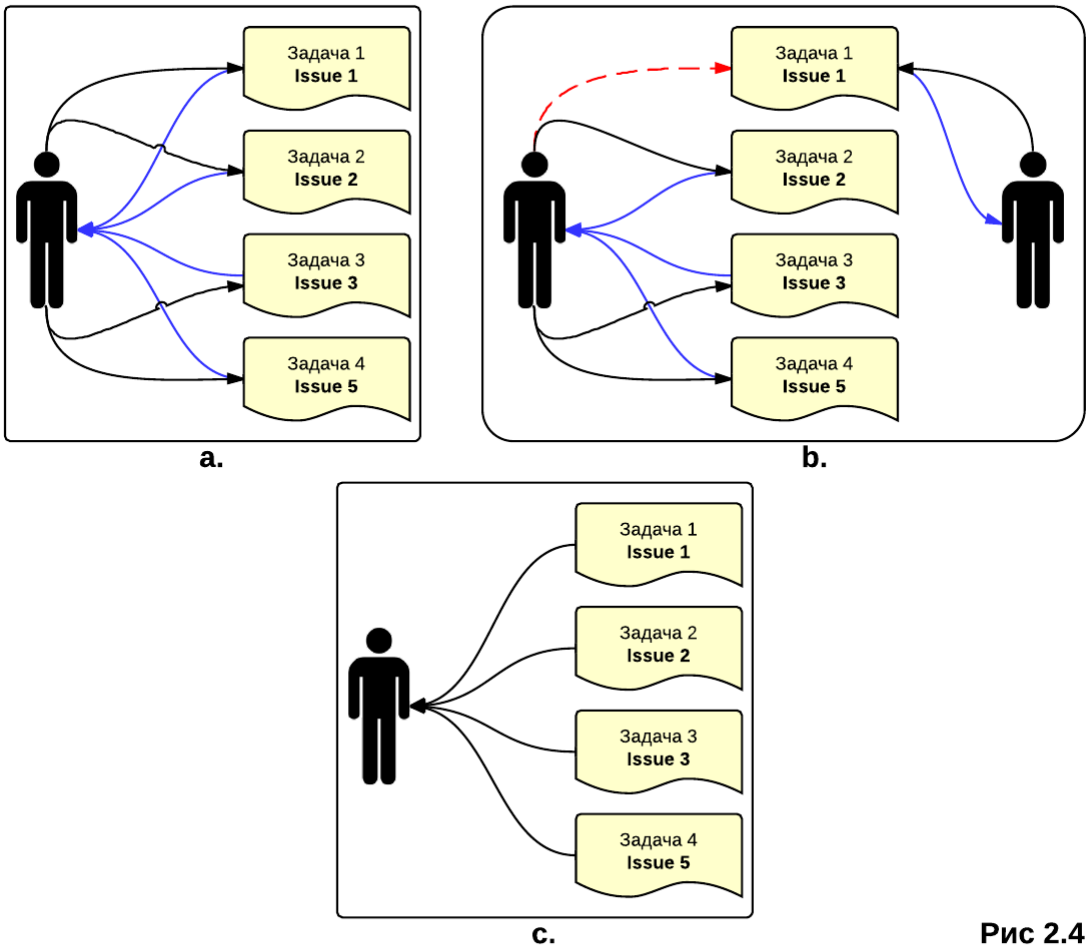

Рис 2.4

Получается, при каждом переназначении задачи, нужно будет также удалять и ссылку на переназначенную задачу из документа *“Исследователь”*. Получается, что нам нужно совершать дополнительное, и, как мы рассмотрим на примере, указанном на **Рис.2.4.c**, не нужное действие.

В примере на **Рис.2.4.c**, в документе *“Задача”* также хранятся ссылки на документ *“Исследователя”*, которому она назначена, но, отсутствуют обратные ссылки. Как не трудно догадаться, при переназначении задачи другому исследователю, у нас не останется не каких лишних ссылок, которые нужно будет повторно удалять.

В итоге, подход, приведенный на **Рис.2.4.c**, не только уменьшает количество связей

между документами, но и убирает вероятность нахождения ложных ссылок, таких как на **Рис.2.4.b**. Именно поэтому решено хранить связи между исследователями и задачами, назначенными им, в одностороннем порядке.

Структура документа *“Задача”* приведена в **Листинге 2.2**.

```
        {
    _id: <Issue-ID>, // идентификатор задачи
research: <Research-ID>,// идентификатор исследования в котором создана задача
from: <Researcher-ID>,// идентификатор того, кем создана задача
to: <Researcher-ID>,// идентификатор того, кому назначена задача
name: <Issuename>,   // название задачи
desc: <Issuedescription>,// описание задачи
date_start: <Dateofissue>,    // дата начала выполнения/создания задачи
deadline: <Datawhenshouldbeend>,// дата, когда задача должна быть завершена
date_closed: <Datewhenissuereallyend>,// дата, реального завершения задачи
priority: <Priority-level>,// приоритет задачи
closed: <bool>,          // закрыта ли задача
db_section: {// секция с флагами базы данных
        _id: <Section-type-ID>,
sec_data:{
deleted: <Bool>,
date_created: <date>,
date_modified: <date>
        }
    }
}
```

Листинг 2.2. Модель документа *“Задача”*

2.3.3 Разграничение прав доступа.

Требования

Не все исследователи должны иметь одинаковые возможности доступа и изменения данных испытуемых. К системе приводятся следующие требования по разделению этих самых возможностей, или прав доступа к данным испытуемых:

1. В зависимости от прав доступа исследователя, данные испытуемых должны предоставляться ему либо полностью, либо в обезличенном виде. То есть таким образом, чтобы нельзя было определить личность испытуемого
2. Доступ к коллекции (или базе данных) всех испытуемых имеют только исследователи, обладающие особым привилегированным статусом. Остальные

же имеют доступ к данным только тех испытуемых, которые принимают участие в тех же исследованиях, что и сам исследователь.

3. В зависимости от прав доступа, исследователь имеет или не имеет возможность вносить изменения, в уже занесенные, данные испытуемых.
4. В зависимости от прав доступа, исследователь имеет или не имеет возможность добавлять новые данные в документы испытуемых.
5. Исследователь не может назначать задачи старшему исследователю.
6. Исследователь может назначать задачи сам себе.

Роли

Разграничение прав производится по средством присвоения, каждому исследователю, различных ролей. *“Роль”* представляет собой документ, в котором хранится набор прав доступа, присвоенный ей. Структура *“Роли”* приведена в **Листинге 2.3.**

```
{
        _id : <Role-ID>,
priority : <Int>
        privileged : <BOOL>,
individuality : <BOOL>,
        read_only : <BOOL>,
        append : <BOOL>,
}
```

Листинг 2.3. Модель документа *“Роль”*

В соответствии со структурой, каждая роль имеет следующий набор полей:

- **_id**- идентификатор роли. Здается явно, при добавлении роли.
- **priority** - приоритетность роли. Нужно чтобы разграничить права распределения задач.
- **privileged** - логическое поле, указывает на то, обладает ли исследователь привилегированным статусом. YES - да обладает и соответственно имеет доступ к коллекции (или БД) всех испытуемых), NO - нет, только через исследование.
- **individuality** - логическое поле, указывает на то, в обезличенном ли виде, исследователю, имеющему данную роль, предоставляются данные испытуемых. YES - данные предоставляются без изменений, NO - в обезличенном виде.

- **read_only** - логическое поле, указывает на то, имеет ли право исследователь, имеющий данную роль, вносить изменения в уже занесенные данные испытуемых.
- **append** - логическое поле, указывает на то, имеет ли право исследователь, имеющий данную роль, добавлять новые данные к документам испытуемых.

В документе *“Исследователь”*, хранится не сам документ *“Роль”*, а ссылка на данный документ. Сами же *“Роли”* хранятся в отдельной коллекции. Данный подход позволяет, в случае чего, легко изменять права доступа целой группы исследователей, имеющих определенную роль. При изменение или добавление каких либо данных к определенной роли, нужно будет произвести изменения лишь в одном документе, а не строить сложные запросы для произведения данного изменения в каждом документе *“Исследователь”*.

2.3.4 Дополнительно: Словарь.

Так как в будущем возможно внедрение проекта для ведения международных исследований. То стоит создать определенную структуру словаря для трансляции слов, записанных в базе данных в указанный язык. Также, после того как пользователем будут добавлены новые секции в документ *“Испытуемый”*,потребуется аналогично хранить имена этих секций, для отображения на стороне клиента.

Например: пользователь создал секцию под названием “Психологический опрос”, внутри БД секция будет называться “ts_psihoOpros”. При считывание клиентом информации из БД ему нужно будет корректно отобразить название секции и всех полей, содержащихся в ней, в удобном для человека формате на требуемом языке. Здесь и встает вопрос об использовании словаря.

```
{
_id : <Dictionary-language>,
dict_data : {
        db_word1 : <lang-word>,
        db_word2 : <lang-word>,
        db_word3 : <lang-word>,
        ...
        db_wordN : <lang-word>
    }
}
```

Листинг 2.4. Модель документа *Словаря*

В **Листинге 2.6** приведен пример предлагаемой модели словаря. Словарь представляет собой специализированную секцию. В поле **_id** хранится специализированное название языка словаря, которое и является его идентификатором, например: ru_ru, для русского и us_eng, для английского. Поле **dict_data**, представляет собой документ, поля которого сопоставляются следующим образом: **db_word** - слово в формате базы данных, то есть названия поля в ней. Значением данного поля является перевод данного слова на язык словаря.

2.3.5 Листинги основных объектов.

```
        {
_id : <Researcher-ID>,                    // идентификатор исследователя, генерируется случайно
rer_main_section : {                       // секция с основными данными
_id : <Section-type-ID>,
sec_data : {
last_name   : <'Фамилия'>,
first_name  : <'Имя'>,
third_name  : <'Отчество'>,
sex         : <Bool>,           //true - мужской, false - женский
age         : <'Возраст'>,
                    dBirth: <'Дата рождения'>
job         : <'Место работы'>
        }
    },
rer_resources_section : {                  // ресурсная секция
_id : <Section-type-ID>,
sec_data : {
equipment : [                      // массив оборудования
                {
eq_name : <'Название оборудования'>,
eq_desc : <'Описание оборудования'>
                },
                ...,
                {
eq_name : <'Название оборудования'>,
eq_desc : <'Описание оборудования'>
                },
],
methods : [// массив исследовательских методов
                {
md_name : <'Название метода'>,
```

```
md_desc : <'Описание метода'>
                },
                ...,
                {
md_name : <'Название метода'>,
md_desc : <'Описание метода'>
                },
],
        }
    },
rer_busyness_section : {            // бизнес секция
_id : <Section-type-ID>,
sec_data : {
status  : <'Текущий статус(занят; свободен...'>,
        role : <Role-ID>
        }
    },
rer_db_section : {// секция с флагами базы данных
_id : <Section-type-ID>,
sec_data : {
        login : "Login",             // Логин исследователя
        pass : "Password hash",      // Хэшкод пароля для входа в систему
deleted         : <Bool>,
date_created    : <date>,
date_modified   : <date>
        }
    }
}
```

Листинг 2.4. Модель документа *“Исследователь”*

```
        {
_id : <TestSubject-ID>,                    // идентификатор испытуемого, генерируется случайно
ts_main_section : {                         // секция с основными данными
_id : <Section-type-ID>,
sec_data :  {
last_name   : <'Фамилия'>,
first_name  : <'Имя'>,
third_name  : <'Отчество'>,
anonym      : <Bool>,              // если значение true, то вышеописанные поля пусты и
sex         : <'Пол'>,               // не рассматриваются
age         : <'Возраст'>,
        dBirth      : <'Дата рождения'>
        }
    },
ts_0_section : {                           // пользовательская секция
```

```
_id : <Section-type-ID>,
sec_data :  {
data1 : <Some-Data>,
data2 : <Some-Data>,
data3 : <Some-Data>,
            ...
dataN : <Some-Data>,
        }
    },
    ...
ts_N_section : {
_id : <Section-type-ID>,
sec_data :  {
data1 : <Some-Data>,
data2 : <Some-Data>,
data3 : <Some-Data>,
            ...
dataN : <Some-Data>,
        }
    },
db_section : {// секция с флагами базы данных
_id : <Section-type-ID>,
sec_data :  {
deleted         : <Bool>,
date_created    : <date>,
date_modified   : <date>
        }
    }
}
```

Листинг 2.5. Модель документа *“Испытуемый”*

```
        {
_id : <Research-ID>,                        // идентификатор исследования, генерируется случайно
res_main_section : {                        // секция с основными данными
_id : <Section-type-ID>,
sec_data : {
name : <Research-Name>,
goal : <Research-Goal>,
initiator : <Researcher-ID>,
        active : <Bool>// активно ли исследование или же оно уже закончено
        date_closed : <Date when research has been end>
        }
    },
researhers : [<Researcher-ID>, <Researcher-ID>, ..., <Researcher-ID>],
testsubjs  : [<Testee-ID>, <Testee-ID>, ..., <Testee-ID>],
```

```
db_section : {// секция с флагами базы данных
_id : <Section-type-ID>,
sec_data :  {
deleted         : <Bool>,
date_created    : <date>,
date_modified   : <date>
        }
    }
}
```

Листинг 2.6. Модель документа *“Исследование”*

Глава 3

В данной главе рассмотрены методы взаимодействия с документами, хранящимеся в БД. Рассмотрены типовые запросы к БД. Также рассмотрено то, каким образом производится визуализация процесса распределения задач связанных с обработкой психофизиологических данных. Подробности реализации пользовательского интерфейса решено не включать в данную, работу, так как данная задача является довольно тривиальной.

§3.1 Добавление, удаление, изменение и чтение данных в системе.

В данном параграфе рассмотрены методы добавления, удаления, обновления и чтения (запроса) основных и вторичных документов в БД. Описаны алгоритмы действий системы при определенных действиях пользователя.

3.1.1 Добавление.

Для добавления объекта в БД в MongoDB используется функция **insert()**. Данная функция может быть вызвана только у коллекции, например: **db.collection.insert()**. Где, **db** - имя базы данных, содержащей нужную коллекцию; **collection** - имя коллекции, в которую требуется вставить документ. Если документы добавляется в несуществующую коллекцию, то такая коллекция создается с вставленными в нее документами. Более подробно о синтаксисе функции **db.collection.insert()** можно узнать из [8].

При добавлении нового довумента в коллекцию, MongoDB автоматически добавляет ему поле **_id** со случайным значением ObjectId(...), если только это поле не указано явно, при добавлении. ObjectId() - это функция создающая объект id размером 12 байт. Как правило в нее подается специально сгенерированная строка из 12 символов.

Исходя из Главы 2, мы имеем семь коллекций: три коллекции основных документов: *researchers* (исследователи), *testsubjs* (испытуемые), *researches* (исследования) ; три коллекции побочных документов: *sections* (секции), *permissionRoles* (роли), *issues* (задачи); одну дополнительную коллекцию словарей: *dicts*. На данном этапе, все коллекции находятся в одной БД, под названием rmsDB (ResearchesManagementSystemsDatabase). Далее рассмотрены методы добавления объектов в каждую коллекцию, используя стандартный

mongoshell, с описанием правильной последовательности добавления и причин именно такой последовательности добавления объектов.

Секции.

Как известно из §2.3, все основные объекты в нашей БД состоят из секций. Так как, предусмотрена возможность добавления пользовательских секций, то все типы секций (иначе говоря каркасы) хранятся в отдельной коллекции *sections*. Изначально данная коллекция заполняется базовыми (основными) типами секций: основные секции испытуемого, исследователя и исследования; секция БД. Имена данных секций хранятся в поле **_id** и задаются, администратором, явно, при добавлении в коллекцию. Имена пользовательских секций генерируются на уровне клиента, исходя из имени, указанного пользователем при создании новой секции.

Например, пользователь создает секцию, под названием, *"Psychologicalsurvey"*, по-умолчанию в приложении используется английский язык. Далее, в зависимости от размера каждого слова, берется часть от его начала и слова соединяются между собой, по следующему правилу: каждое новое слово, кроме первого, начинается с большой буквы - в итоге получается: *"psySur"*. Далее, к названию добавляется префикс *"ts_"* (от TestSubject) и окончание *"_section"*. В итоге получается уникальный идентификатор: *"ts_psySur_section"*. Полученное значение также добавляется в английский словарь в соответствии: *"ts_psySur_section"* : *"Psychologicalsurvey"* (добавление данных рассматриватся в разделе 3.1.2). По такаму же принципу генерируюся и пользовательские поля, содержащиеся в **sec_data**, но уже без добавления префикса и окончания. Данные поля, заполнятся значениями "по-умолчанию" введенными пользователем при создании, или проще говоря, их названиями.

В **Листинге 3.1** показан пример добавления основной *"Секции"* испытуемого в коллекцию *sections*:

```
db.sections.insert({
_id : "ts_main_section",
sec_data : {
last_name : "Last Name",
first_name : "First Name",
middle_name : "Middle Name",
anonym      : Boolean(false),    // таким образом в MongoDB задаются значения типа Boolean
sex : Boolean(true),
age : "Age",
dBirth : Date()    // Данная конструкция задает текующую дату, для создания даты
```

```
})                          // рождения используется newDate(). Например newDate(Jun 8, 1984)
```

Листинг 3.1. Добавление “Секции”

Необязательно вставлять только один документ за раз, функция **insert()** также умеет выполнять вставку массивов документов, следующим образом:

```
db.collection.insert( [ {<document1>}, {<document2>}, …, {<documentN>} ] )
```

Листинг 3.2. Синтаксис добавления массивов документов функцией insert()

Мало вероятно, что такой подход понадобится при выполнении пользовательских запросов, но он может быть полезен при заполнении, администратором, начальных данных и добавлении новых словарей в БД.

Исследователи. Роли.

Исходя из модели *“Исследователя”*, представленной разделе 2.3.5, каждому исследователю, при добавлении, назначается роль, определяющая его права доступа к системе. Следовательно, прежде чем добавлять исследователя в БД, у нас уже должны существовать хоть какие-то роли, для назначения исследователю. В **Листинге 3.2** рассмотрен пример добавления роли лаборанта в коллекцию *permissionRoles*.

```
db.permissionRoles.insert({
_id : "per_low",                    // имя роли, для словаре
priority : 1,
privileged : Boolean(false),
individuality : Boolean(false),
read_only : Boolean(false),
append : Boolean(false)
})
```

Листинг 3.3. Добаление “Роли” лаборанта

Поле **_id**, у роли, также как и у секции, задается явно, при ее добавлении в коллекцию. На данном этапе не требуется возможность добавления пользовательских ролей и какой-либо привязки роли к конкретному исследованию. Каждая роль имеет абсолютное значение и права доступа каждого исследователя одинаковы, независимо от исследования. Каждая новая роль добавляется только администратором.

После добавления в БД, нужного количества ролей и необходимых секций испытуемого, можно приступать к добавлению самого документа *“Испытуемый”*. Пример

добавления *“Испытуемого”* показан в **Листинге 3.4**.

```
db.researchers.insert({
{
rer_main_section : {
_id: "rer_main_section",
sec_data : {
last_name: "Ivanov",
first_name: "Ivan",
middle_name: "Ivanovich",
sex: Boolean(true),
age: 45,
dBirth: new Date("May 2, 1969"),
job: "SRI FFM, Psihometrical Lab"
            }
        },
rer_resources_section : {
_id : "rer_resource_section",
sec_data : {
equipment : [
                    {
eq_name : "MRI",
eq_desc : "Magnetic Resonance Imaging"
                    }
                ],
methods : [
                    {
md_name : "MRI",
md_desc : "Magnetic Resonance Imaging"
                    }
                ]
            }
        },
rer_busyness_section : {
_id : "rer_busyness_section",
sec_data : {
status  : "Free",
role : "per_low"
            }
        },
db_section : {
_id : "db_section",
sec_data : {
login : "IvIvIv",
pass : "0sns0a10sks919s9d9192dx9ws01",
deleted         : Boolean(false),
```

```
        date_created    : Date(),
        date_modified   : Date()
                    }
                }
            }


        })
```

Листинг 3.4. Добавление “Исследователя”

Стоит заметить, что поля **_id**, при добавлении исследователя не указывается и, следовательно, генерируется автоматически средствами MongoDB.

Добавлять новых исследователей в систему имеют право только администраторы и привилегированные пользователи (исследователи с привилегированной ролью). Администратор, как правило, выполняет это через mongoshell. Ну а пользователь пользуется предоставляемой формой в клиенте.

Исследования. Задачи. Испытуемые.

В целом, для добавления исследования используется тот же принцип, что и для исследователя. Синтаксис не отличается ничем, кроме структуры документа и названия коллекции (*researches*). Создание исследования происходит через специальную форму. Исследователь, создавший исследование, автоматически, становится его участником.

Добавить новую задачу пользователь может через специальную форму. Для добавления задачи, должно существовать хотя бы одно активное исследование. Пользователь, добавляющий задачу, автоматически, становится ее инициатором и лишь он может переназначить ее кому-либо другому. Сиснтаксис mongoshell, для добавления задачи, также аналогичен сиснтаксису добавления исследователя, меняется лишь название коллекции (*issues*) и сама структура документа.

Добавление испытуемых отличается от добавления исследователей лишь дополнительной возможностью создания новых секций через специальную форму. Процесс генерирования новой секции был описан в разделе 3.1.1 *Секции*.

Словари.

Как описано в разделе 2.3.4, словари являются отдельными документами. Как и

все предыдущие документы словари храняться в своей отдельной коллекции *dicts*. На данном этапе добавлением новых словарей в БД может заниматься только администратор. Для этого ему нужно не только заполнить уже имеющиеся в других словарях поля. Но и обеспечить возможность добавления новых слов в этот словарь. Так как на данном этапе не требуется обеспечение работы с несколькими словарями, точный способ обеспечения записи новых слов в множество словарей на требуемом в каждом словаре языке здесь не описаны. Примеры того, каким образом можно попробовать обеспечить данную функцию, приведены в разделе 3.1.2 *Словарь* .

3.1.2 Изменение и удаление.

Удаление.

Для удаления документов в MongoDB существует функция **db.collection.remove()**. Но, при наличие большого количества документов и множества связей между ними, такая операция может оказаться довольно сложной и, при ошибке, может привезти к образованию ложных ссылок на уже удаленные документы. К тому же всегда существует вероятность того, что документ был удален по ошибке, от этого никто и никогда не застрахован.

Для предотвращения такого рода ошибок, в §2.3, в моделях всех основных документов и документа *“Задача”* предусмотрена, так называемая, “Секция базы данных”. В этой секции содержится логическое поле **deleted**. В случае, если пользователь (исследователь), обладает достаточными правами доступа, чтобы удалить какой-либо документ, он, со своей стороны, просто жмет на кнопку “Удалить”. И для него все удалено. Но, на самом деле ничего не удаляется, а лишь изменяется флаг базы, со значения *false* на *true*, и при запросах на чтение такие документы попросту игнорируются.

Возможность удаления документы, не имеющих “Секции базы данных”, таких как: *“Секция”*, *“Роль”* и *“Словарь”* пользователю не предоставляется. Данная операция может проводиться лишь администратором, и лишь случаях крайней необходимости, например, при наличие множества неиспользуемых секций или ролей.

Для изменения, добавления и удаления полей и массивов в документах, в MongoDB, используется функция **db.collection.update()**. Данная функция имеет множество операторов для обеспечения нужных операций. Подробнее о синтаксисе функции и ее операторах можно узнать из [9].

Для того чтобы удалить, добавленного ранее, пользователя, следует операцию, показанную в **Листинге 3.5**.

```
db.researchers.update(
    { "_id" : ObjectId("538492fc343431654886f039") },
    {
$set : {
            "rer_db_section.sec_data.deleted" : Boolean(true),
            "rer_db_section.sec_data.date_modified" : Date()
}
    }
)
```

Листинг 3.5. Пример удаления документа “Исследователь”

В данном случае, в первой секции фигурных скобок, указывается принцип выбора документов, в которых будет производиться изменение. В данном случае выбор производится по полю **_id**, которое было установлено случайным образом и имеет значение ObjectId("53848f64343431654886f037"), следовательно будет выбран только один документ. В следующей секции фигурных скобок, указывается оператор, в данном случае это оператор **$set**. Данный оператор производит изменения значения полей, на заданные, а в случае, если поля не существует, создает поле с указанным значением. Во внутренней секции оператора **$set**описываются поля с новыми знеачениями. В данном случае выбирается два поля внутри набора данных секции **rer_db_section**: **deleted** и **date_modified**. Поле **date_modified**, должно всегда обновляться при изменении любой информации в документе, включая, как в данном случае, его удаление.

Стоит отметить, что в, приведенном в §2.3, формате представления данных в БД, только секция базы данных испытуемого имеет особое название, у остальных документов она называется просто **db_section**.

Удаление поля документа не рассматривается, так как пользователю не нужно будет, удалять поля, из уже созданных секции или уже заполненные секций из документов “Испытуемых”, по причине того, что после удаления какого-либо поля по ошибке, его уже будет довольно трудно восстановить. Стоит лишь сказать, что это производится с помощью оператора **$unset**, подробнее о котором, можно узнать из [9].

Изменение.

Поле

Как уже было оговорено в предыдущей секции, изменение полей в

MongoDBпроизводится с помощью функции **db.collection.update()**. В **Листинге 3.6**, приведен синтаксис изменения поля в документе с произвольным “_id”, произвольной секции, произвольной коллекции.

```
db.someCollection.update(
    { "_id" : "Some-ID" },
    {
$set : {
            "some_section.sec_data.someField" : "<New-value>"
}
    }
);
```

Листинг 3.6. Синтаксис изменения поля в секции произвольного документа

Для изменения какого-либо поле, содержащегося не в секции, а в самом документе, напмер *“Задаче”*, нужно лишь указать название этого поля и новое значение. То есть убрать из **Листинга 3.6**: *some_section.sec_data* - и оставить лишь *someField*.

В **Листинге 3.7** приведен пример переназначения задачи другому исследователю.

```
db.issues.update(
    { "_id" : "Some-ID" },
    {
$set : {
            "to" : "Some-ID",
            "db_section.sec_data.date_modified" : Date()
        }
    }
)
```

Проверка того, может ли данный исследователь переназначать задачу и может ли он назначить эту задачу конкретному исследователю, производится на уровне клиента. Алгоритм действий системы, при различных действиях пользователя, следующий:

Действие пользователя (исследователя)	Действия системы
а) Исследователь запрашивает данные задачи на просмотр	1. Отображается тело задачи 2. Значение поля **from**,

	запрашиваемой задачи, сравнивается с полем **_id**, запрасившего задачу, исследователя 3. Если поля совпадают, то кнопка “Переназначить”, в интерфейсе, становится активной 4. Иначе: неактивной
б) Исследователь назначает задачу другому исследователю	1. Система сравнивает **_id**, назначающего с назначаемым. Если значения совпадают - задача назначается 2. Иначе. Система запрашивает роли обоих исследователей 3. Если приоритет назначающего задачу выше, чем у назначаемого - задача назначается 4. Иначе. Система выдает сообщение об ошибке.
в) Исследователь переназначает задачу другому исследователю	1. Пункт 2 из случая а). На случай ошибок. 2. Все как в случае б).

Добавление слов в словарь, выполняется простым запросом по полю **_id** коллекции *dicts*, с использованием оператора **$set**.

Объект массива

Также стоит рассмотреть способ добавления нового элемента в массив. Как известно, в документе *“Исследование”* находится два массива, массив исследователей, *researchers* и массив испытуемых, *testsubjs*. В каждом из массивов хранятся ссылки (идентификаторы), на объекты из одноименных коллекций. Допустим, требуется добавить какого либо исследователя к исследованию, или же наоборот, исключить какого либо убрать какого-либо испытуемого из исследования. Для добавления и удаления объектов в массивы, у

функкции **update()**, существуют операторы **$addToSet** и **$pull**. В **Листинге 3.8** показаны примеры добавления нового исследователя к исследованию и удаления, из него, испытуемого.

```
db.researches.update(
    { "_id" : "Some-research-ID"},
    {
$addToSet : {
            "researhers" : "Some-researcher-ID"}
        }
    }
);


db.researches.update(
    { "_id" : "Some-research-ID" },
    {
$pull : {
            "testsubjs" :"Some-test-subject-ID"
}
    }
)
```

Листинг 3.9. Пример добавления идентификатора исследователя в массив исследователей и удаления идентификатора испытуемого из массива испытуемых в исследовании

Добавление новой секции в документ “Испытуемый”

Для добавления новой секции, требутся лишь, полностью заполненная пользователем, через форму, секция. Конечно пользователь может не заполнять в ней некоторые поля, тогда они будут добавлены в пустом виде. Допустим, что существует секция *“ts_pregData_section”*, в которой хранятся некоторые данные о беременности женщины. В **Листинге 3.10** показан пример добавления такой секции к произвольному испытуемому.

```
db.testsubjs.update(
    { "_id" : "Some-ID" },
    {
$set : {
            "ts_pregData_section" : {
_id : "ts_pregData_section",
sec_data : {
gaveBirth : Boolean(true),
```

```
        dateOfGaveBirth : new Date ("March 28, 1993")
                        }
                    },
                    "db_section.sec_data.date_modified" : Date()
        }
            }
        )
```

Листинг 3.10. Пример добавлени секции в произвольный документ “Испытуемый”

Из данного примера можно понять, что добавление новый секции, практически аналогично, изменению какого-либо поля. Только лишь, после указания названия секции нужно вписывать полное тело секции, со всеми заполненными полями.

3.1.3 Чтение.

Для чтения данных в MongoDB используется функция **db.collection.find()**. Подробное описание синтаксиса данной функции можно узнать из [10].

Для того, чтобы получить все документы содержащиеся в коллекции, нужно просто вызвать данную функцию не включая никаких дополнительных аргументов.

Для того, чтобы найти какой либо документ, например по его полю **_id**, используется та же конструкция, что и для поиска документа в функции **db.collection.update()**.

В **Листинге 3.11** показан пример получения **_id** и всех данных из основной секции всех исследователей, находящихся в коллекции *researchers*. Данный запрос нужен при назначении участников исследования.

```
        db.researchers.find(
            { },
            {
                "rer_main_section.sec_data" : 1
        }
        );
```

Листинг 3.11. Пример получения основных данных и идентификаторов всех исследователей

В первом блоке фигурных скобок указывается по каким полям будет проходить выборка, если ничего не указано, то выбираются все документы из секции. Во втором блоке, записываются поля, по следующему правилу: если напротив поля стоит 1, то оно будет отображено в результатах поиска; если 0, то нет; если указано несколько полей, напротив которых стоит 1, то все неуказанные поля не отображаются в результатах поиска,

исключением является только поле **_id**, оно всегда отображается в результатах; для того, чтобы поле **_id** не отображалось в результатах, следует вписать его и поставить на против 0.

Запрос списка исследователей с ограничениями

Теперь рассмотрим следующий вариант: пользователь хочет добавить в исследование новых участников. В данном случае, система не должна предлагать ему добавить тех участников, которые уже учавствуют в нем. Для начала, нужно получить список всех участников, уже учавствующих в исследовании, это делается простым запросом по полю *researchers* (пример показан в **Листинге 3.12**).

```
db.researches.find(
    { "_id" : "Some-research-ID" },
{
        {
            "researhers" : 1,
            "_id" : 0
        }
    }
)
```

Листинг 3.12. Запрос массива идентификаторов исследователей, участвующих в исследовании

Далее система обрабатывает ответ и генерирует запрос в виде, приведенном в **Листинге 3.13**.

```
db.researchers.find(
    {
$and : [
            {
_id : { $ne : "Some-researcher-ID"  }
            },
            {
_id : { $ne : "Some-researcher-ID"  }
            }
        ]
    },
    { "rer_main_section.sec_data" : 1 }
)
```

Листинг 3.13. Пример запроса идентификаторов и данных основной секции всех

```
                    исследователей, кроме двух, чьи идентификаторы указаны в условиях
```

В данном примере используется два, ранее не описанных, оператора, **$and** и **$ne**. Оператор **$and** представляет собой логическое “И”, которому на вход подается массив условий. Оператор **$ne** представляет собой логическое “НЕ РАВНО”. Словесно пример, приведенный в **Листинге 3.13**, может звучать так: найти всех исследователей, у которых поле **_id** не равно следующим значениям. В примере показано лишь двое, но точно также, через запятую можно указать и всех остальных.

Запрос роли исследователя и обезличенных данных испытуемого

Теперь стоит рассмотреть вопрос предоставления данных испытуемого исследователю. Как описано в разделе 2.3.3, в зависимости от прав доступа исследователя, данные испытуемого должны предоставлять ему полностью, либо в обезличенном виде.

Рассмотрим алгоритм получения данных испытуемого в виде, соответствующем правам доступа исследователя:

1. Запрашиваем идентификатор роли у документа *“Исследователь”* из коллекции *researchers*. Как правило на этот момент данные испытуемого уже загружены из БД в систему, но имеет смысл рассмотреть данный вопрос.

```
db.researchers.find(
    { "_id" : "Some-researcher-ID"},
    {
        "rer_busyness_section.sec_data.role" : 1,
"_id" : 0
    }
)
```

Листинг 3.14. Запрос идентификатора роли исследователя

2. Запрашиваем поле *individuality* по идентификатору роли из коллекции *permissionRoles*

```
db.permissionRoles.find(
    { "_id" : "Some-role-ID" },
    {
        "individuality" : 1,
        "_id" : 0
    }
```

```
)
```

Листинг 3.14. Запрос поля individuality из документа “Роль”

3. В зависимости от значения поля *individuality*. Запрашиваем данные испытуемого. Если значение false, то запрашиваем обезличенные данные. Исключаются ФИО, дата рождения и адрес.

```
db.testsubjs.find(
    { "_id" : "Some-testSubject-ID"},
    {
        "ts_main_section.sec_data.last_name" : 0,
        "ts_main_section.sec_data.first_name" : 0,
        "ts_main_section.sec_data.middle_name" : 0,
        "ts_main_section.sec_data.dBirth" : 0,
        "ts_main_section.sec_data.address" : 0
}
);
```

Листинг 3.14. Запрос обезличенных данных испытуемого

4. Иначе, простой запрос по идентификатору пользователя.

Запрос списка всех задач назаченых исследователю

Данный запрос является достаточно тривиальным. Нужно лишь сделать запрос в коллекции *issues* по одному полю **to**, в котором следует указать **_id** исследователя, задачи которого требуется найти. Пример запроса приведен в листинге **3.15**.

```
db.issues.find(
    {
        "to" : "Some-researcher-ID"
}
);
```

Листинг 3.15. Запрос на получения всех задач назначенных исследователю

Аналогичным запросом, но по полю **from**, можно получить список всех задач, которые были назначены исследователем. Также выполнив запрос по полю **research** и подав **_id**какого-либо исследования, можно получить список всех задач в данном исследовании.

Для получения всех задач, назначенных какому-либо исследователю, в рамках

какого-либо исследования, необходимо выполнить запрос приведенный в **Листинге 3.16**.

```
db.issues.find(
    {
        "to": "Some-researcher-ID",
        "research": "Some-research-ID"
    }
);
```

Листинг 3.16. Запрос всез задач назначенных исследователю в рамках исследования

§3.2 Визуализации процесса распределения задач связанных с обработкой психофизиологических данных.

Визуализацию процесса распределения задач связанных с обработкой психофизиологических данных предложено реализовать в виде ориентированного графа, узлами которого являются исследователи, а связи показывают от кого, кому поступило задание.

Для визуализации используется язык JavaScript, используются средствами JavaScript-библиотеки dagre-d3. Для отображения данных библиотека dagre-d3 использует средства более мощной библиотеки D3.js. Для отображение данных в библиотеке dagre-d3 используется формат JSON, а как уже известно именно в таком формате мы получаем данные из нажей БД. Подробнее о возможностях данных библиотек можно узнать из [11, 12].

На **Рис. 3.1** показан пример данного графа.

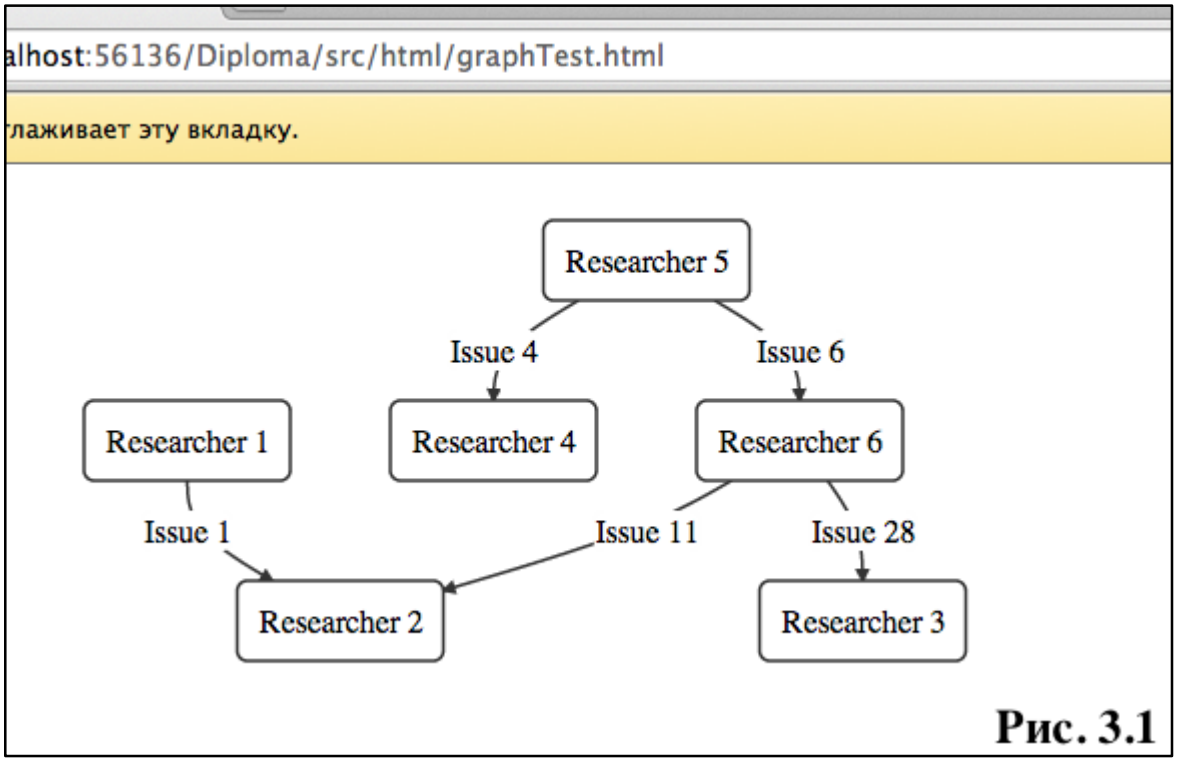

Рис. 3.1

Как видно из **Рис. 3.1**, на графе также дополнительно подписываются имя задачи, которая была назначена исследователю. Стоит заметить, что отображается только название последней задачи, назначенной одним исследователем другому. В случае, если задача, назначена исследователем самому себе, то это также будет отражено на графе, процесс показан на **Рис 3.2**.

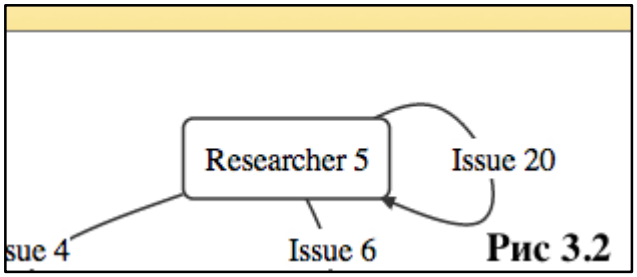

Рис 3.2

Также , граф реагирует на наведение мыши на узел или заголовок задачи, выделяя его, примеры на **Рис 3.3**.

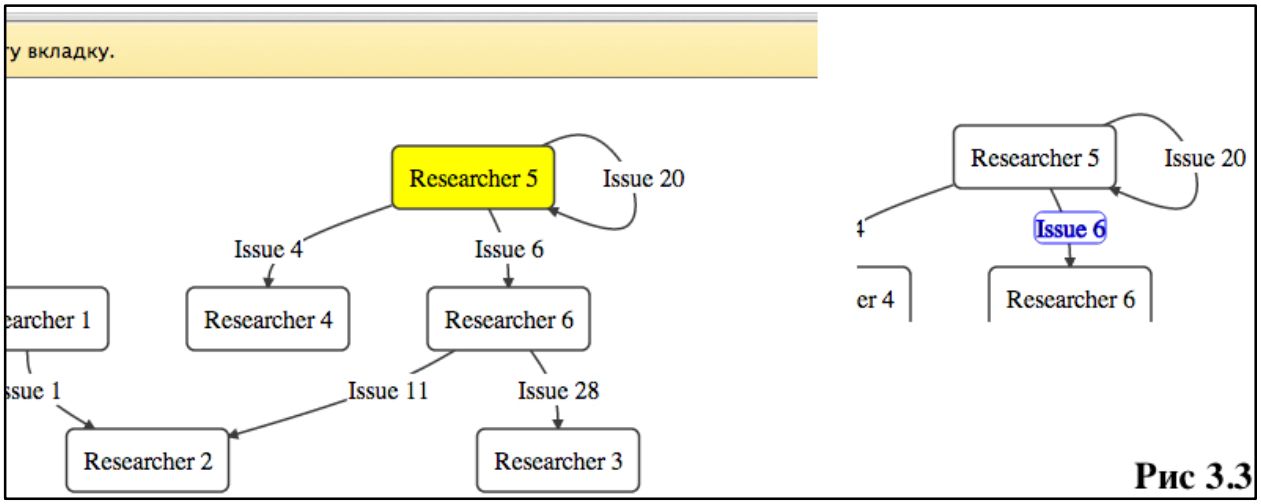

Рис 3.3

При нажатие на узел (исследователя), открывается профиль этого исследователя со списком всех назначенных ему задач.

Стоит сразу отметить, что, при отображении графа, на данном этапе, не делалась ставка на красивый дизайн, так как была лишь задача корректно визуализировать сам процесс распределения задач.

Запрос на отображение графа может быть произведен лишь из страницы исследования. При запросе на отображения графа, система выполняет следующие действия:

1. Запрашивает из БД список всех исследователей, принимающих участие в данном исследование.
2. Добавляет их в граф, как узлы. Пример:

```
graph.addNode("Researcher-string-ID", { label : "Researcher last and first names"})
```

3. Для каждого исследователя запрашивает список всех задач, назначенных в рамках данного исследования
4. Выбирает из этих задач, задачу с самой свежей датой (последнюю назначенную)
5. Формирует список этих задач
6. Добавляет задачи в граф, как связи. Пример:

```
graph.addEdge("Issue-string-ID", "from Researcher-string-ID", "to Researcher-string-ID", { label : "Issue short name" })
```

Заключение.

В настоящее время в России выполняется масштабная модернизация здравоохранения. В частности, одним из основных направлений, является внедрение медицинских информационных систем в практику работы государственных учреждений здравоохранения. К примеру, в США, практически каждый медицинский центр или больница уже довольно давно имеют компьютерную линию, позволяющую медицинским работникам иметь доступ к базе данных пациентов.

Научно-исследовательская информационная система, предназначенная для автоматизации процесса постановки исследовательских задач, связанных с обработкой психофизиологических данных испытуемых, является специализированной разработкой регионального сегмента Единой государственной информационной системы в сфере здравоохранения (ЕГИСЗ) в соответствии с «Концепцией информатизации», и федеральным законом №323-ФЗ.

Список источников.

1. “Информационные технологии в медицине” Под. ред. Г.С. Лебедева, Ю.Ю. Мухина. М: Издательство «Радиотехника», 2012 г. – 208 стр.: ил. ISBN 978-5-88070-327-2
2. http://www.consultant.ru/document/cons_doc_LAW_156802/ - Федерального закона Российской Федерации "Об информации, информационных технологиях и о защите информации" от 27 июля 2006 года № 149-ФЗ
3. http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=7229 - ISO/IEC 2382-1:1993 Information technology -- Vocabulary -- Part 1: Fundamental terms
4. http://intexpro.ru/Doc/StandartIS-Zdrav.doc - стандарт организации СТО МОСЗ 91500.16.0002-2004 «Информационные системы в здравоохранении. Общие требования», при участии Лебедева Г.С., Емелина И.В. и Столбова А.П.
5. http://medical.nema.org/standard.html - DICOM standart
6. http://jsman.ru/mongo-book/ - The Little MongoDB Book by Karl Seguin rus
7. http://docs.mongodb.org/manual/ - документация MongoDB
8. http://docs.mongodb.org/manual/reference/method/db.collection.insert/ - MongoDB, функция db.collection.insert
9. http://docs.mongodb.org/manual/reference/operator/update/ - MongoDB, операторы функции db.collection.update()
10. http://docs.mongodb.org/manual/reference/method/db.collection.find/ - MongoDB, функция db.collection.find()
11. https://github.com/cpettitt/dagre-d3 - библиотека для визуализации графа
12. http://d3js.org - официальный сайт библиотеки D3.js
13. Медицинская информатика : Учебник / И.П. Коро- люк. – 2 изд., перераб. и доп. – Самара : ООО «Офорт» : ГБОУ ВПО «СамГМУ». 2012.— 244 с; ил.