Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»

(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:

«Создание многофункционального сайта кафедры ИУ-9»

Студент ИУ9-62 _____ А.А. Олохтонов
(Подпись, дата)

Руководитель курсового проекта _____ И.Э. Вишняков
(Подпись, дата)

Москва, 2018 г.

Содержание

Введение

На данный момент сайт кафедры ИУ-9 [1] является скорее подробной аннотацией, чем полноценным веб-порталом. Учитывая, что текущая страница iu9.bmstu.ru занимает первое место среди результатов запроса «кафедра иу9» в четырех самых популярных поисковых системах России (Яндекс, Google, go.mail.ru и Рамблер [2, 3, 4, 5]), для многих она становится первым знакомством с кафедрой.

Более того, сайт кафедры потенциально может выполнять более одной функции. Так, для абитуриентов и прочих заинтересованных лиц он предоставляет справочную информацию о поступлении и кафедре в целом, а для студентов и преподавателей может служить универсальным средством односторонней и двусторонней связи, организации и проведения кафедральных мероприятий.

Целью данного курсового проекта является написание нового сайта кафедры ИУ-9. Задачи, таким образом, включают: анализ страниц и оформления существующего сайта; перенос информации на новый сайт; создание недостающих страниц; разработку строгого единого стиля; разработку требований к новому функционалу сайта и построение модели данных на основании этих требований; выбор программных средств и реализацию нового серверного и клиентского функционала; тестирование полученной реализации на работоспособность и удобство использования.

1 Разработка функционала

Для удобства весь функционал сайта был разделен на три логические части:

1. статические страницы, предназначенные для просмотра всеми пользователями и не содержащие интерактивных элементов;
2. статические страницы с динамическим наполнением, содержание которых определяется типом учетной записи пользователя;
3. интерактивные страницы, позволяющие взаимодействовать с сервером при помощи кнопок, текстовых полей и т.п.

Очевидно, что для второго и третьего пунктов необходимо разработать систему регистрации и авторизации, причем регистрацию пользователей нужно ограничить: пользователь может лишь оставить заявку, а окончательное решение о добавлении учетной записи принимает администратор сайта.

Упомянутое одностороннее взаимодействие было решено реализовать в виде новостной ленты, где для каждой статьи можно указать целевую аудиторию. Также предусмотрена возможность создания кафедральных событий: преподаватель может выбрать тип события, заполнить описание, прикрепить файлы и указать участников, которые получат уведомления.

Для разграничения доступа была разработана ролевая модель, в которой каждому пользователю приписывается одна или несколько ролей, которые определяют доступный этому пользователю функционал сайта.

1.1 Статические страницы

Существующий сайт кафедры ИУ-9 содержит три страницы: «главная», «абитуриенту» и «наука». В новой версии сайта эти страницы сохранены, однако ин-

формация для абитуриентов дополнена и разбита на три отдельных страницы: «поступление», «наши выпускники» и «учебный план». Также добавлена страница «новости». Для навигации в верхней части всех страниц было добавлено меню, содержащее ссылки на страницы и кнопку «войти».

1.1.1 Главная странциа

Главная страница по сути единолично составляет первое впечатление о сайте (а для многих и о кафедре), поэтому ей стоит уделить особое внимание. В новой версии сайта главную страницу решено было упросить относительно существующей версии, оставив лишь три небольших блока: «о кафедре», «особенности подготовки» и «отличия от других вузов и кафедр».

1.1.2 Наука

Оригинальная версия страницы с научной деятельностью уже содержит всю необходимую информацию, поэтому оказалось достаточно привести её к более читаемому виду, чего удалось достичь по сути автоматически, применив к ней новый единый стиль.

1.1.3 Новости

Страница новостей играет двойную роль: для читателей — это новостная лента, содержание которой определяется данными учетной записи (либо её отсутствием в случае незарегистрированных посетителей сайта); для пользователей с более высоким уровнем доступа страница должна содержать форму написания новости и способ редактирования уже отправленных сюжетов.

Также на странице новостей решено было выводить краткую информацию по событиям, в которых данный пользователь принимает участие. Каждое событие

оформлено в виде ссылки на страницу с более подробной информацией.

1.1.4 Поступление

Так же, как и страница «наука», оригинальная версия этой страницы содержит всю необходимую информацию, поэтому остается лишь перенести эту информацию на новый сайт и применить к ней новую таблицу стилей.

1.1.5 Наши выпускники

Одна из страниц, которой нет на оригинальном сайте кафедры, это «наши выпускники», однако её стоит добавить как минимум по двум причинам. Во-первых, трудоустройство выпускников — для многих решающий фактор при выборе места обучения. Во-вторых, крайне высокий процент выпускников кафедры ИУ-9 работает по специальности, что также встречается нечасто и заслуживает упоминания.

1.1.6 Учебный план

Учебный план был доступен для скачивания со старой версии сайта кафедры [6], но впоследствии удален. На новой странице «учебный план» решено представить план в двух версиях: оформленный в читаемом виде и разбитый по курсам и семестрам; в виде таблицы, доступной для скачивания в формате pdf.

1.2 Страницы с динамическим наполнением

Страницы с динамическим наполнением подразумевают реализацию с двух сторон: со стороны клиента (то есть клиентского JavaScript кода) и со стороны сервера. С клиентской стороны необходимо скрыть те элементы, которые пользователь видеть не должен и запросить данные с сервера, чтобы заполнить оставши-

еся элементы нужной информацией. С серверной стороны необходимо опознать от какого пользователя получен запрос, для того чтобы отправить клиенту только ту информацию, к которой у него есть доступ. Такая валидация необходима, потому что клиент может «представиться» совсем не тем, кем является на самом деле, изменив клиентский JavaScript код или cookie [7] встроенными средствами браузера или сделав запрос к серверу напрямую, к примеру при помощи утилиты curl [8].

Примером страницы с динамическим наполнением являются упомянутая ранее новостная лента, содержание которой целиком определяется уже после загрузки статической части страницы. Однако в процессе разработки функционала публикации и редактирования новостей возникла идея универсального встраиваемого модуля, который бы позволял редактировать любой текст на сайте прямо из браузера.

1.2.1 Editbox

Такой встраиваемый элемент должен позволять редактировать блоки текста в формате Markdown [9], гораздо более удобным для человека, чем html. Также предусмотрена возможность предпросмотра и сохранения текстов. Для того чтобы редактирование можно было продолжить с другого компьютера, исходные коды должны сохраняться на сервере и подгружаться при нажатии кнопки «редактировать». Более того, для пользователей, которым не доступен функционал редактирования, необходим отдельный запрос на получение уже преобразованных блоков. Идентификатором блока будет служить его id на странице. С клиентской стороны нужно реализовать интерфейс для редактирования, предпросмотра и сохранения кодов в markdown.

1.3 Интерактивные страницы

Интерактивными назовем страницы, подразумевающие взаимодействие клиента с сервером. Чаще всего такое взаимодействие реализуется при помощи различных кнопок и форм ввода, к которым привязаны различные запросы к серверу.

Интерактивными элементами сайта кафедры таким образом будут формы авторизации и регистрации, форма публикации новостей, вышеупомянутый editbox, административная страница «управление» и страница проведения события. Помимо взаимодействия с сервером также необходимо производить валидацию вводимых данных со стороны клиента, чтобы минимизировать возможность ошибок в отправляемых данных.

1.3.1 Публикация новостей

Форма публикации новостей состоит из текстового поля для заголовка и поля для тела новости. Для выбора целевой аудитории предусмотрен выпадающий список, позволяющий выбрать студентов с точностью до группы, преподавателей и всех. Чтобы упростить редактирование уже опубликованных новостей, каждый элемент новостной ленты состоит из заголовка, мета-информации (автор новости, дата и время) и editbox с телом новости.

1.3.2 Управление

Страница управления содержит весь административный функционал: создание событий, обработка заявок на регистрацию, а также добавление пользователей напрямую (заявку на можно оставить только для студенческой учетной записи). При этом создание событий доступно всем преподавателям, а остальной функционал — только администратору сайта. Обработка заявок позволяет принять или отклонить заявку. Прямая регистрация пользователей позволяет задать фамилию,

имя, логин, пароль и роль нового пользователя, после чего учетная запись будет мгновенно создана, не попадая в список заявок.

1.3.3 Регистрация и авторизация

Для пользователя, не прошедшего авторизацию, в верхнем меню отображается кнопка «войти». Нажатие на неё открывает форму авторизации. Для авторизованного пользователя отображается кнопка «выйти». В случае успешной авторизации на сервере заводится сессия и её идентификатор устанавливается в cookie браузера, с которого эта авторизация была произведена. Сессия считается активной в течение суток с последнего успешного запроса.

Форма регистрации позволяет указать фамилию, имя, желаемый логин и пароль, группу, в которой состоит регистрируемый студент. Полученная заявка появляется на странице управления. Введенный логин и пароль становятся пригодными для авторизации только после подтверждения заявки. В случае если логин или пароль введены неверно — отображается сообщение об ошибке.

1.4 Ролевая модель для доступа

Так как многие запросы к серверу должны быть доступны только для определенных пользователей, было решено ввести ролевую модель доступа [10]. А именно, при регистрации каждому пользователю приписывается некоторая роль. Для каждой из ролей определены набор разрешенных и запрещенных запросов, и на основании этих наборов каждый запрос проверяется на корректность. Если обнаруживается нарушение — пользователь получает вместо данных отказ в доступе. Так как сессия, содержащая информацию о роли пользователя, хранится со стороны сервера, такая модель обеспечивает довольно надежное разграничение доступа к данным.

1.4.1 Посетитель

Незарегистрированный посетитель не входит в ролевую модель, однако требует упоминания. Дело в том, что большинство страниц, доступных для таких пользователей, статические, то есть не содержат дополнительных запросов к серверу, а потому не требуют и проверки. Исключение составляют запросы на получение содержание editbox и публичных новостей, а также запрос на авторизацию.

1.4.2 Студент

Роль студента предусматривает базовый функционал: получение новостей, предназначенных для той группы, в которой состоит данный студент и участие в событиях (включает в себя получение данных о событии и отправку сообщений в чат события).

1.4.3 Преподаватель

Роль преподавателя полностью включает в себя роль студента (для преподавателя разрешены все запросы, разрешенные студенту). Помимо этого, для преподавателя доступны запросы создания события и сопутствующие вспомогательные запросы: получение существующих студенческих групп, зарегистрированных преподавателей, списка предметов и типов мероприятий. Также для преподавателя доступен запрос на завершение события.

1.4.4 Редактор

Роль редактора также целиком включает все возможности студента. Дополнительный функционал, доступный редактору: получение исходных кодов editbox, и запрос на их обновление. Также редактор может публиковать и редактировать новости.

1.4.5 Администратор

Администратор сайта имеет доступ ко всем запросам без какого-либо ограничения. То есть в отличии от студента, преподавателя и редактора, администратор может отклонять и принимать заявки на регистрацию и добавлять новых пользователей напрямую.

1.5 Обмен сообщениями

Для удобного многостороннего общения в рамках одного события предусмотрена возможность чата, то есть обмена мгновенными сообщениями. Причем необходимо реализовать не только отправку и получение (и отображение) сообщений, но и получение истории переписки для данного чата. Так как такой чат может быть прикреплен к событию, необходимо предусмотреть возможность создания чат-комнаты с конкретным набором участников. Тогда пользователь может присоединиться к конкретному чату, и все отправленные сообщения будут приходить только участникам этого же чата. Также необходимо предусмотреть возможность закрытия чат-комнаты (например, когда событие завершилось).

1.6 События

Событие описывается названием, предметом, типом события, датой и временем его проведения. Участниками события могут быть группы студентов и конкретные преподаватели. Также к событию можно прикрепить файлы, то есть загрузить их на сервер и прикрепить гиперссылку к событию. Опционально, на время существования события можно создать чат-комнату, доступную только участникам события.

1.6.1 Создание и завершение события

При создании события нужно указать тип и предмет. Если одного или их обоих нет в выпадающем списке — можно выбрать пункт «другой» и ввести нужное значение в текстовом поле. После создания события новый тип (предмет) сохранится и будет доступен в виде пункта выпадающего списка при создании всех последующих событий. Также все его участники должны получить уведомление (на странице новостей).

После того как преподаватель завершил событие, оно должно пропасть из новостей всех участников. В случае, если студент попробует пройти на страницу события по прямой ссылке, он должен получить информацию о том, что событие удалено. Чат, прикрепленный к завершенному событию должен закрыться, то есть не принимать и не рассылать сообщения.

2 Реализация

2.1 Модель данных

Учитывая описанные требования, имеет смысл включить в модель данных следующие сущности: пользователь, новость, элемент editbox, роль, заявка на регистрацию, чат-комната, событие, тип и предмет события. Тогда ER-диаграмма принимает следующий вид:

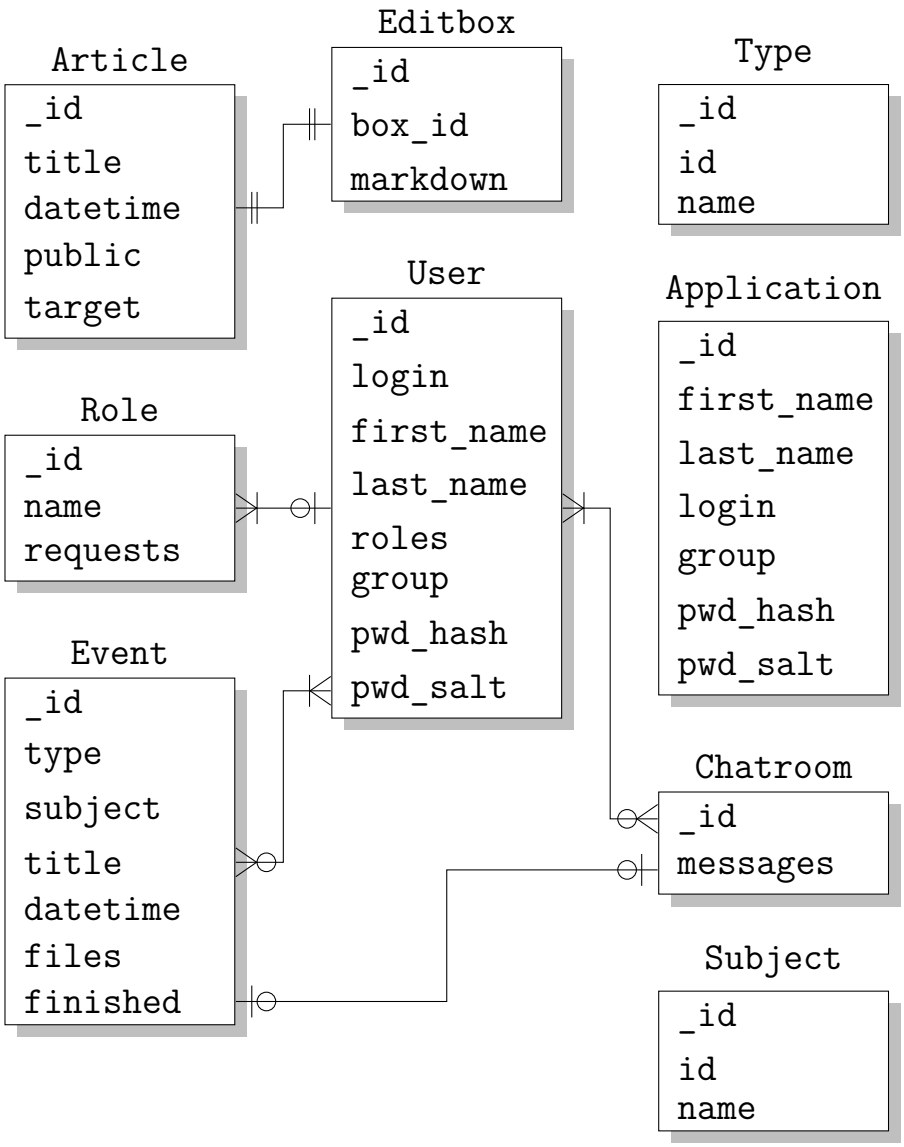

Схема 1 — ER-модель

2.2 Выбор технологий

Для разработки клиентской части сайта использовался JavaScript без каких-либо дополнительных библиотек (за исключением одного файла showdown.min.js [11], используемого для предпросмотра изменений editbox). Такое решение было принято для уменьшения размера скачиваемого клиентом кода и, как следствие, ускорения загрузки страниц.

Разработку сервера было решено вести на Node.js [12] с использованием фреймворка express, так как этот набор инструментов предоставляет удобный компромисс между скоростью разработки и быстродействием написанного кода. Более того, написание серверного кода на Node JS позволяет легко изменять и дополнять функционал при помощи встроенного пакетного менеджера npm [13]. В качестве СУБД была выбрана документо-ориентированная MongoDB. Причиной такому выбору послужили, в основном, два факта: относительная простота реализуемой модели данных и удобство работы с JSON в JavaScript приложениях.

В целях повышения производительности было также решено использовать СУБД Redis, являющуюся высокопроизводительным хранилищем типа “ключ-значение” [14]. Redis используется для хранения данных, к которым требуется регулярный доступ в процессе работы сервера. К примеру, такой информацией являются сессии пользователей и таблица ролей.

2.2.1 MongoDB

В СУБД MongoDB данные представляются в виде BSON [15] (супермножество JSON) документов и объединяются в коллекции. В данном курсовом проекте предусмотрены следующие коллекции:

`users` — коллекция зарегистрированных пользователей. Каждый документ содержит: фамилию, имя, логин, массив названий ролей и хеш и соль пароля.

В случае, если одна из ролей пользователя — студент, то пользователь также содержит поле group, содержащее значащую часть индекса группы студента (н: 12 или 71);

`roles` — возможные роли пользователей. Документ содержит название роли и словарь, где ключем является конкатенация через пробел метода и пути HTTP запроса (н: `GET /users/applications`), а значением — еще один словарь. Этот словарь на данный момент содержит только один ключ — поле `available`, значением же является `true` или `false`. Таким образом, каждая операция с сервером для данной роли может быть запрещена или разрешена. В дальнейшем в словаре можно хранить еще и JSON-схему, накладывающую произвольные ограничения на параметры запроса;

`applications` — коллекция заявок на регистрацию. Структура документов по сути идентична документам из коллекции `users`, за тем исключением, что заявки на регистрацию не содержат поле `role`;

`chatrooms` — коллекция чат-комнат. Каждый документ содержит историю переписки и массив логинов участников чата;

`editbox` — коллекция содержания элементов editbox. Документы содержат два поля: код в markdown и поле `box_id`, соответствующее id HTML элемента, содержание которого определяется данным документом (id хранятся в base64 [16])

`events` — коллекция событий. Документ содержит сохраненные данные из формы создания события, такие как название события, название предмета и типа события и т.д. Также, событие содержит поле `files`, являющееся словарем, где ключем выступает отображаемое имя прикрепленного файла, а

значением — относительный путь к файлу на сервере. Таким образом по названию прикрепленного файла можно получить прямую ссылку на сам файл. Событие может содержать поле `chatId`, совпадающее с одним из `ObjectId` в коллекции `chatrooms`. Наличие этого поля означает, что к событию прикреплена чат-комната. Наконец, событие может содержать флаг `finished`, устанавливаемый в том случае, когда событие завершилось;

`eventTypes` — коллекция доступных типов событий (н: лабораторная работа). Эта коллекция необходима для формирования выпадающего списка «тип события». Если при создании списка указан тип «другой», то происходит поиск по коллекции `eventTypes`, и если тип с таким названием не найден, то он добавляется в коллекцию (и становится доступен для выбора при создании последующих событий);

`eventSubjects` — коллекция доступных предметов при создании события. Полностью идентична коллекции `eventTypes` по структуре документов;

`news` — коллекция новостей. Каждая новость содержит заголовок, отображаемое имя автора, дату создания новости, поле `public` либо `target`, задающее целевую аудиторию новости и поле `boxId`, устанавливающее соответствие с документом в коллекции `editbox`, содержащим тело новости.

Для ускорения операций над коллекциями MongoDB и обеспечения гарантий уникальности были введены следующие индексы:

- уникальный индекс по полю `login` в коллекции `users`;
- уникальный индекс по полю `id` в коллекциях `eventTypes` и `eventSubjects`;
- уникальный индекс по полю `box_id` в коллекции `editbox`;

- индекс по полю `group` в коллекции `users` для быстрой агрегации и поиска пользователей;

- индекс по полю `roles` в коллекции `users` для быстрой выборки по ролями (н: найти всех преподавателей);

- индекс по полям `public` и `target` в коллекции `news` для быстрого поиска новостей.

2.3 Архитектура сервера

Исходный код сервера был классическим образом разделен на четыре части:

1. файл `app.js`, содержащий код конфигурации и запуска сервера;

2. каталог `routes`, содержащий код, отвечающий за выбор нужного обработчика для запроса и подключение связующего программного обеспечения (англ: middleware). Разделение запросов на требующие и не требующие авторизации было сделано при помощи создании двух роутеров, только к одному из которых было подключено middleware проверки сессии.

3. каталог `handlers`, содержащий код обработчиков для запросов, причем обработчики различных классов запросов описаны в отдельных файлах. К примеру, все запросы, относящиеся к работе с пользователями, обрабатываются в файле handlers/users.js. Такие запросы обычно начинаются с некоторого префикса (в случае с пользователями: /users/…). С целью повторного использования кода был также написан файл handlers/common.js, содержащий часто используемые в обработчиках функции, такие как ответ кодом 200, 400, 404 или 500 с опциональной передачей сообщения клиенту.

4. каталог `lib`, содержащий код самописных библиотечных функций, будь то функции работы с базами данных или методы валидации сессий и паролей. Для удобства функции работы с Redis и MongoDB были описаны в двух файлах: lib/mongo_client.js и lib/redis_cilent.js, производящих инициализацию соответствующих адаптеров (в момент импортирование модуля). Файл lib/mongo_client.js отличается от lib/redis_client.js тем, что в первом описаны функции, выполняющие конкретные запросы к MongoDB, в то время как lib/redis_client.js содержит только обертку для самих методов Redis API. Также был написан файл lib/db.js, предоставляющий абстракцию для работы с данными.

2.4 Страницы

Так как было принято решение не использовать фреймворков для написания клиентского JavaScript кода, были реализованы некоторые утилитарные функции, среди которых:

функция `get_cookies()`, составляющая словарь с данным, хранящимися на момент вызова функции в cookie браузера:

```
function get_cookies() {
  let cookies = {}, pairs = document.cookie.split(";")
  for (let pair of pairs) {
    pair = pair.split("=")
    if (pair.length == 2) { cookies[(pair[0] + '').trim()] = unescape(pair
    [1]) }
  }
  return cookies
}
```

Листинг 1 — функция извлечения cookie

функции `css_get()` и `css_set()`, позволяющие получить или изменить указанный набор css свойств объекта:

```
function css_set(element, properties) {
  if (typeof element === 'string') {
    element = document.getElementById(element)
  }

  Object.keys(properties).forEach((key) => {
    element.style[key] = properties[key]
  })
}

function css_get(element, property) {
  if (typeof element === 'string') {
    element = document.getElementById(element)
  }

  const style = window.getComputedStyle(element)
  return style.getPropertyValue(property)
}
```

Листинг 2 — функции получения и изменения таблицы стилей

функции `logged_in()` и `logged_in_as()`, осуществляющие проверку наличия нужной роли в cookie клиента:

```
function _logged_in_as(role) {
  let ui_view = []
  if ('UIVIEW' in COOKIES) { ui_view = COOKIES['UIVIEW'].split(',') }
  else { return false }
  return ui_view.indexOf(role) != -1
}
```

Листинг 3 — функция клиентской проверки авторизации

функция `get_dim()`, возвращающая ширину и высоту переданного элемента, что необходимо для замены текстового поля editbox на textarea соответствующего размера:

```
function get_dim(element) {
  const rect = element.getBoundingClientRect()
  return {'width': Math.round(rect.width), 'height': Math.round(rect.height)}
}
```

Листинг 4 — функция получения размеров элемента

функция `request()`, представляющая обертку для асинхронного AJAX [17] запроса и принимающая в качестве параметра функцию-callback:

```
function _request(method, url, headers, data, callback, isFile) {
  if (METHODS.indexOf(method) != -1) {
    let xhttp = new XMLHttpRequest()
    xhttp.onreadystatechange = () => {
      if (xhttp.readyState === 4)
        callback(xhttp.status, xhttp.responseText)
    }

    if (method === 'GET' && data !== null) {
      let keys = Object.keys(data)
      url += ('?' + keys[0] + '=' + data[keys[0]])
      for (let i = 1; i < keys.length; ++i)
        url += ('&' + keys[i] + '=' + data[keys[i]])
    }

    xhttp.open(method, SERVER + url, true)
    if (headers !== null) {
      Object.keys(headers).forEach((key) =>
        xhttp.setRequestHeader(key, headers[key]))
    }
    if (method == 'GET' || data === null) xhttp.send()
```

```
22      else xhttp.send(isFile ? data : JSON.stringify(data))
23    } else {
24      callback(400, 'unknown method: ' + method)
25    }
26  }
```

Листинг 5 — функция-обертка асинхронной отправки запроса

Так как значительная часть клиентского JavaScript кода отвечает всего лишь за скрытие и отображение соответствующих элементов, были написаны два простейших css класса `initially-hidden` и `initially-disabled`, которые впоследствии добавлялись или удалялись из списка классов объекта.

2.5 Авторизация

Для реализации механизма авторизации была выбрана схема сессионности, лежащая между классической схемой с передачей данных авторизации в каждом запросе и технологией JWT (JSON Web Token [18]). Суть метода заключается в следующем: с запросом на авторизацию пользователь передает свой логин и пароль, которые проверяются на сервере. В случае успешной проверки на сервере (в данном случае в Redis) создается сессия. Значение в Redis в данном случае является сокращенным документом из коллекции users, а ключ — уникальным идентификатором, генерируемым в момент открытия сессии. Этот уникальный идентификатор записывается клиенту в cookie вместе с его ролью, чтобы с обеспечить «сессионность» с клиентской стороны:

```
1  function login(req, res) {
2    const user = basic_auth(req)
3
4    if (!user) {
```

```
    common.send_bad_request_response(res)
  }

  db.get_user(user.name, (err, user_db) => {
    if (err) {
      common.send_error_response(res, 'could not find user: ' + err.message)
      return
    }

    if (!user_db || !security.checkPassword(user.pass, user_db.passwordSalt,
    user_db.passwordHash)) {
      common.send_unauthorized_response(res, 'user not found or the password is
    incorrect')
      return
    }

    const session_id = uuid()
    db.open_session(session_id, user_db, (err) => {
      if (err) {
        common.send_error_response(res, 'could not open session: ' + err.message)
        return
      }

      res.cookie('SESSIONID', session_id, {httpOnly: true, maxAge: 86400 * 1000})
      res.cookie('UIVIEW', user_db.roles.join(','), {httpOnly: false, maxAge: 86400
    * 1000})
      common.send_text_response(res, 200)
    })
  })
}

function open_session(session_id, user_data, callback) {
  let user_copy = JSON.parse(JSON.stringify(user_data))
  delete user_copy['_id']
  delete user_copy['passwordHash']
  delete user_copy['passwordSalt']

  redis_client.set(session_id, JSON.stringify(user_copy), (err) => {
```

```
    if (err) {
      callback(err)
    } else {
      redis_client.expire(session_id, 86400, callback)
    }
  })
}
```

Листинг 6 — обработка запроса авторизации и открытие сессии

Все последующие запросы, произведенные с идентификатором сессии в cookie, позволяют таким образом быстро получить информацию о пользователе, просто считав значение из Redis по ключу. Проверка авторизации упрощается до проверки считанного из Redis значения на null.

Время жизни сессии установлено в 86400 секунд (24 часа) с последнего успешного запроса (то есть сессия автоматически продлевается).

Запрос на завершение сессии, соответственно, удаляет сессию из Redis и очищает cookie для отражения изменений со стороны клиента.

2.6 Ролевая модель

Для обеспечения ролевой модели коллекция `roles` была заполнена документами, отражающими описание каждой из ролей. Для проверки доступности запроса было написано middleware, осуществляющее валидацию всех запросов, требующих валидации:

```
function access_check_middleware(req, res, next) {
  const session_id = req.cookies.SESSIONID

  if (!session_id) {
    common.send_unauthorized_response(res)
    return
```

```
  }

  db.get_session(session_id, (err, user_db) => {

    if (err) {
      common.send_unauthorized_response(res, err.message)
      return
    }

    req.user_db = user_db
    req.session_id = session_id

    validation.check(req, (err) => {
      if (err) {
        common.send_forbidden_response(res, err.message)
      } else {
        next()
      }
    })
  })
}

function check(request, callback) {
  const user = request.user_db
  const req_uri = request.method + ' ' + new url.URL('http://' + request.headers.
   host +  request.url).pathname

  db.get_roles(user['roles'], (err, roles) => {
    if (err) {
      callback(err)
      return
    }

    if (!roles) {
      callback(new Error('no such roles: ' + user['roles']))
      return
    }

```

```
    for (let role of roles) {
      const rule = role['requests']
      if (req_uri in rule) {
        if (!rule[req_uri]['available']) {
          continue
        } else {
          callback()
          return
        }
      } else {
        continue
      }
    }

    callback(new Error('prohibited'))
  })
}
```

Листинг 7 — проверка роли в middleware

2.7 Сообщения

Для обмена мгновенными сообщениями был написан handlers/websocket.js, отвечающий за инициализацию websocket сервера и поддержание инфраструктуры чат-комнаты: сохранение истории переписки и рассылку сообщений участникам. Идентификатор чат-комнаты передается в параметрах url при подключении к websocket серверу. Таким образом, достаточно проверить что подключившийся пользователь авторизован и находится в списке участников данной чат-комнаты.

Установленный протокол поддерживает три типа сообщений:

1. `SINGLE_MESSAGE` — одно сообщение, содержащее присланный текст и логин отправителя;

2. `MESSAGE_HISTORY` — история переписки. В поле `history` содержит массив сообщений в формате `SINGLE_MESSAGE`;

3. `ERROR` — содержит только текст ошибки, без информации об отправителе.

Так как входящие сообщения пользователей рассматриваются как текстовые, необходимо проводить некоторую «чистку» во избежание манипуляций с исходным кодом страницы, на которой эти сообщения будут отображаться. Для достижения этой цели была написана функция `sanitize`, заменяющая все контрольные символы на эскейп-последовательности:

```
function sanitize(str) {
  return (str.replace(/&/g, '&amp;')
    .replace(/"/g, '&quot;')
    .replace(/'/g, '&#x27;')
    .replace(/</g, '&lt;')
    .replace(/>/g, '&gt;')
    .replace(/\//g, '&#x2F;')
    .replace(/\\/g, '&#x5C;')
    .replace(/`/g, '&#96;'))
}
```

Листинг 8 — очистка текста сообщений от служебных символов

Для ускорения работы сервиса информация о чат-комнате выгружается в локальный словарь при первом успешном обращении. Отображение логина пользователя в websocket объект также записывается в локальный словарь.

2.8 События

Помимо подгрузки доступных типов/предметов при создании события необходимо также получить списки преподавателей и учебных групп. Для получения

списка преподавателей достаточно сделать запрос по коллекции `users` с фильтром по роли. Получение студенческих групп производится при помощи чуть более сложного запроса, использующего оператор `$aggregate`:

```
function get_user_groups(callback) {
  do_client_action((client) =>
    client.collection(usersCollectionName)
    .aggregate([{'$match': {'group': {'$exists': true}}},
    {'$group': {'_id': '$group', 'count': {'$sum': 1}}}]).toArray(callback),
    callback)
}
```

Листинг 9 — получение групп студентов

В свою очередь, добавление участников в событие добавляет на самом деле не самих пользователей, а «правила», которые при отправке события на сервер преобразуются в реальный список пользователей при помощи функции `unwind_rules`:

```
function unwind_rules(rules, callback) {
  let match_query = {'$or': []}
  rules.forEach((rule) => {
    match_query['$or'].push(rule)
  })

  do_client_action((client) =>
    client.collection(usersCollectionName)
    .find(match_query)
    .project({'_id': 0, 'login': 1, 'first_name': 1, 'last_name': 1, 'roles': 1})
    .toArray(callback), callback)
}
```

Листинг 10 — развертка праавил для получения списка участников

Наконец, для упрощения операции получения событий, в которых участвует конкретный пользователь, при создании события оно дописывается в документы

всех соответствующих пользователей. При удалении же события, записи удаляются и из документов, соответствующих участникам.

3 Тестирование

Тестирование работоспособности производилось вручную: проверками всех сценариев работы сайтом и сравнением реального поведения с ожидаемым. Проверенные сценарии представлены в виде таблицы:

Описание сценария	Ожидаемое поведение	Реальное поведение
Проход по статическим ссылкам, доступным для неавторизированного пользователя	Браузер переходит на соответствующую страницу	Соответствует ожиданиям
Переход по ссылке новости, будучи авторизованным как студент или преподаватель	В ленте новостей видны публичные новости и новости для группы (преподавателей)	Соответствует ожиданиям
Переход по ссылке новости, будучи авторизованным как редактор или администратор	Помимо новостной ленты видна форма публикации новости	Соответствует ожиданиям
Заполнение формы и отправка новости	Страница перезагружается, новость появляется во всех соответствующих в лентах	Соответствует ожиданиям

Нажатие на элемент editbox, будучи авторизованным за редактора или администратора	Появляется интерфейс редактирования, состоящий из двух кнопок	Соответствует ожиданиям
Нажатие на кнопку с пиктограммой карандашика	После загрузки текст editbox заменяется на textarea с исходными кодами в markdown	Соответствует ожиданиям
Нажатие на пиктограмму с глазиком	Исходные коды в markdown заменяются на размеченный и стилизованный текст в HTML	Соответствует ожиданиям
Нажатие на пиктограмму с дискетой	Страница перезагружается, изменения элемента editbox сохраняются	Соответствует ожиданиям
Нажатие на кнопку «ВХОД» после ввода верного логина и пароля	Страница сайта изменяется в соответствии с ролью, надпись «Войти» меняется на «Выйти»	Соответствует ожиданиям
Нажатие на кнопку «ВХОД» после ввода *неверного* логина и пароля	Под формой авторизации появляется красный текст «Неверный логин и/или пароль»	Соответствует ожиданиям

Нажатие на кнопку «Выйти»	Страница возвращается к виду, доступному для неавторизованного пользователя, надпись «Выйти» меняется на «Войти»	Соответствует ожиданиям
Нажатие на кнопку «Создать» в элементе создания нового события	Страница перезагружается. В случае если были указаны нестандартные тип или предмет, они добавляются в соответствующие выпадающие списки	Соответствует ожиданиям
Нажатие на кнопку «Принять» или «Отклонить» на странице управления	В случае успешного добавления/удаления пользователя заявка исчезает из списка. В противном случае выводится сообщение об ошибке	Соответствует ожиданиям
Нажатие на кнопку «Добавить» после заполнения формы создания нового пользователя на странице управления	Страница перезагружается, данные нового пользователя позволяют войти на сайт	Соответствует ожиданиям

Отправление сообщения в чат, в котором состоит данный пользователь	Сообщения появляется в истории переписки всех пользователей данной чат-комнаты	Соответствует ожиданиям
Нажатие на кнопку «Оставить заявку» на странице регистрации	Пользователь перенаправляется на главную страницу, заявка на регистрацию появляется на странице управления	Соответствует ожиданиям

Таблица 1 — ожидаемое и реальное поведение сайта

Заключение

В ходе выполнения курсовой работы были проанализированы информация и её оформление на существующем сайте кафедры ИУ-9 и создан новый, многофункциональный сайт, предоставляющий информацию о кафедре. Был разработан новый единый стиль, в рамках которого были оформлены старые и новые страницы сайта.

На основании поставленных требований был разработан новый функционал, включающий в себя систему авторизации и регистрации, ролевую модель доступа, систему обмена мгновенными сообщениями, публикацию и получение новостей. Разработан встраиваемый модуль, допускающий редактирование страниц сайта в markdown, предпросмотр и сохранение изменений. Также была сформулирована и реализована модель «событий», позволяющая создать и провести кафедральное событие произвольного характера.

Разработанный функционал был реализован с использованием технологии Node JS и СУБД MongoDB и Redis. Клиентская часть исходного кода была написана без использования фреймворков, таким образом уменьшив объем скачиваемых данных на странице в среднем до 250КБайт.

Полученная реализация была протестирована при помощи ручной проверки всех сценариев, найденные неточности поведения исправлены.

Дальнейшее развитие проекта можно направить на развитие ролевой модели, добавление возможности изменения событий после их создания, использование механизма обмена мгновенными сообщениями для создания чата вида «вопрос-ответ».

Список литературы

[1] *Главная страница* // Кафедра ИУ-9 Теоретическая информатика и компьютерные технологии — 2018 — Режим доступа: http://iu9.bmstu.ru/ (дата обращения 17.10.2018)

[2] *Поисковый запрос «кафедра иу9»* // Яндекс — 2018 — Режим доступа: https://yandex.ru/search/?text=кафедра%20иу9&lr=213 (дата обращения 17.10.2018)

[3] *Поисковый запрос «кафедра иу9»* // Google — 2018 — Режим доступа: https://www.google.ru/search?q=кафедра+иу9 (дата обращения 17.10.2018)

[4] *Поисковый запрос «кафедра иу9»* // Мэйл.РУ — 2018 — Режим доступа: https://go.mail.ru/search?q=кафедра+иу9 (дата обращения 17.10.2018)

[5] *Поисковый запрос «кафедра иу9»* // Рамбер — 2018 — Режим доступа: https://nova.rambler.ru/search?query=кафедра%20иу9 (дата обращения 17.10.2018)

[6] *Запрос «http://iu9.bmstu.ru»* // Wayback Machine — 2018 — Режим доступа: https://web.archive.org/web/*/http://iu9.bmstu.ru (дата обращения 17.10.2018)

[7] *HTTP cookie* // Wikipedia - The Free Encyclopedia — 2018 — Режим доступа: https://en.wikipedia.org/wiki/HTTP_cookie (дата обращения 17.10.2018)

[8] *Documentation Overview* // curl - command line tool and library for transferring data with URLs — 2018 — Режим доступа: https://curl.haxx.se/docs/ (дата обращения 17.10.2018)

[9] *Markdown* // Wikipedia - The Free Encyclopedia — 2018 — Режим доступа: https://en.wikipedia.org/wiki/Markdown (дата обращения 17.10.2018)

[10] *Role-based access control* // Wikipedia - The Free Encyclopedia — 2018 — Режим доступа: https://en.wikipedia.org/wiki/Role-based_access_control (дата обращения 17.10.2018)

[11] *A Markdown to HTML converter written in Javascript!* // Showdown — 2018 — Режим доступа: http://showdownjs.com/ (дата обращения 17.10.2018)

[12] *JavaScript runtime built on Chrome's V8 JavaScript engine.* // Node.js — 2018 — Режим доступа: https://nodejs.org/en/ (дата обращения 17.10.2018)

[13] *Package manager for JavaScript* // npm — 2018 — Режим доступа:https://www.npmjs.com/ (дата обращения 17.10.2018)

[14] *Open source (BSD licensed), in-memory data structure store* // Redis.io — 2018 — Режим доступа: https://redis.io/ (дата обращения 17.10.2018)

[15] *Short for Binary JSON* // BSON — 2018 — Режим доступа: http://bsonspec.org/ (дата обращения 17.10.2018)

[16] *Base64* // Wikipedia - The Free Encyclopedia — 2018 — Режим доступа: https://en.wikipedia.org/wiki/Base64 (дата обращения 17.10.2018)

[17] *Ajax (programming)* // Wikipedia - The Free Encyclopedia — 2018 — Режим доступа: https://en.wikipedia.org/wiki/Ajax_(programming) (дата обращения 17.10.2018)

[18] *JSON Web Token* // Wikipedia - The Free Encyclopedia — 2018 — Режим доступа: https://en.wikipedia.org/wiki/JSON_Web_Token (дата обращения 17.10.2018)