МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«**КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Институт среднего профессионального образования**

**(ИНСПО)**

|  |  |
| --- | --- |
|  | ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК  Директор ИНСПО  канд. пед. наук, доцент  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.П. Хлопова  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019г. |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**РАЗРАБОТКА АППАРАТНОЙ ЧАСТИ СИСТЕМЫ МОНИТОРИНГА МЕДИЦИНСКИХ ПОКАЗАТЕЛЕЙ**

Работу выполнил\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н. А. Строкин

(подпись, дата)

Специальность 09.02.03 Программирование в компьютерных системах

Руководитель

Преподаватель ИНСПО \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.Г. Титов

(подпись, дата)

Нормоконтролер

Преподаватель ИНСПО \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Э. С. Егозаров

(подпись, дата)

Краснодар

2019

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc11019189)

[1 Описание используемых технологий 5](#_Toc11019190)

[1.1 Arduino 5](#_Toc11019191)

[1.2 Архитектура микроконтроллеров AVR. 6](#_Toc11019192)

[1.2 Node MCU v.3 на базе ESP8266. 8](#_Toc11019193)

[1.3 Платформа Node JS на языке Java Script 9](#_Toc11019194)

[1.3.1 Семантика и синтаксис 10](#_Toc11019195)

[1.4 Преимущества Node js перед другими платформами для Web–программирования 11](#_Toc11019196)

[1.5 Mongo DB 14](#_Toc11019197)

[1.6 Преимущества Mongo DB 15](#_Toc11019198)

[1.7 Проблемы которые будет решать наш Web–интерфейс. 16](#_Toc11019199)

[2 Разработка серверной части системы мониторинга медицинских показателей 17](#_Toc11019200)

[2.1 Разработка web– интерфейса 17](#_Toc11019201)

[2.5 Структура базы данных 21](#_Toc11019202)

[2.6 Служба поддержки 22](#_Toc11019203)

[ЗАКЛЮЧЕНИЕ 24](#_Toc11019204)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 25](#_Toc11019205)

[ПРИЛОЖЕНИЕ А – Техническое задание 26](#_Toc11019206)

[ПРИЛОЖЕНИЕ Б – Листиннг программ 29](#_Toc11019207)

# ВВЕДЕНИЕ

В последнее время, стараясь улучшить качество и результативность лечения пациентов, требующих постоянного наблюдения, медицинские учреждения нуждаются в более технологичных решениях, которые увеличат скорость диагностики пациента и соответственно приведут к его более быстрому лечению и выздоровлению.

Проект, который мы разработали позволяет реализовывать данную задачу.

Объектом исследования является дистанционный мониторинг медицинских показателей.

Предметом исследования является устройство, позволяющее осуществлять дистанционный мониторинг медицинских показателей, с помощью интегрированных датчиков.

Целью дипломной работы является разработка серверной части автоматизированной системы мониторинга медицинских показателей с выводом данных в глобальную сеть, посредством Web – интерфейса.

Для достижения цели выпускной квалификационной работы поставлены следующие задачи:

* Описать используемые технологии для достижения заданной цели.
* Разобраться в проблеме, и узнать, чего именно не хватает мед. работникам, что облегчило бы их труд.
* Разработать платформу для мониторинга медицинских показателей.

Структура работы обусловлена предметом, целью и задачами исследования. Работа состоит из введения, двух глав, заключения, списка использованных источников и 2 приложений.

Введение раскрывает актуальность, определяет степень научной разработки темы, объект, предмет, цель, задачи.

Первая глава содержит в себе подробную информацию о Микроконтроллерах AVR и базе на которой создавалось устройство, используемых в них датчиках, описание Node MCU v.3 на базе микроконтроллера ESP8266, описание языка Java Script и программной платформы на которой создан Web–интерфейс, node js, сравнение его с другими языками, описание Базы данных использующейся на сервере Mongo DB и сравнение его с другими СУБД, проблемы которые будет решать наш Web–интерфейс, Разработка Дизайна Web–интерфейса.

Вторая глава представляет собой 2 раздела и описывает процесс создания сервера и взаимодействие Web – интерфейса с устройством мониторинга медицинских показателей, используемые в проекте программы.

Заключение включает итоги работы, а также сложности, возникшие в процессе разработки.

В приложениях находятся техническое задание и листинг разработанных микропрограмм.

# 1 Описание используемых технологий

## **1.1 Arduino**

Для удобства разработки и отладки прототипов устройств используются отладочные платы, позволяющие быстро собрать ту или иную схему без пайки за короткий промежуток времени. Некоторые оснащаются программным обеспечением.

Arduino – торговая марка аппаратно–программных средств для построения простых систем автоматики и робототехники. Программная часть состоит из бесплатной программной оболочки (IDE) для написания микропрограмм, их компиляции и прошивки отладочной платы. Аппаратная часть представляет собой набор собранных печатных плат, продающихся как официальным производителем, так и сторонними разработчиками с небольшими модификациями. Полностью открытая архитектура системы позволяет свободно копировать или дополнять линейку продукции. Самая популярная плата – Arduino UNO.

Arduino – представляет собой плату, которая представлена на рисунке 1, с микроконтроллером и процессором ATmega328p. Данный контроллер обладает архитектурой AVR. Данная архитектура обладает рядом особенностей, из которых стоит отметить относительно малую производительность и высокую вибро–температурную устойчивость. Платформа имеет 14 цифровых вход/выходов (6 из которых могут использоваться как выходы ШИМ), 6 аналоговых входов, кварцевый генератор 16 МГц, разъем USB, силовой разъем, разъем ICSP и кнопку перезагрузки. Для работы необходимо подключить платформу к компьютеру посредством кабеля USB, либо подать питание при помощи адаптера AC/DC или батареи.

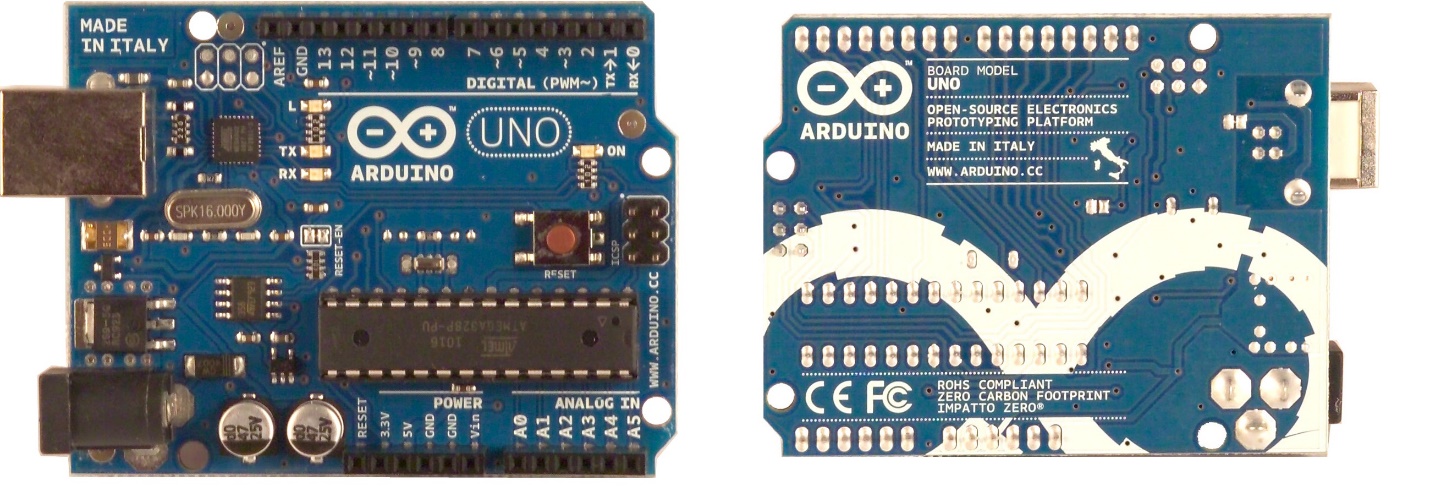


Рисунок 1 – Плата Arduino UNO

Питание. Arduino UNO может получать питание через подключение USB или от внешнего источника питания. Источник питания выбирается автоматически. Платформа может работать при внешнем питании от 6 В до 20 В. При напряжении питания ниже 7 В, вывод 5V может выдавать менее 5 В, при этом платформа может работать нестабильно. При использовании напряжения выше 12 В регулятор напряжения может перегреться и повредить плату. Рекомендуемый диапазон от 7 В до 12 В. Внешнее питание (не USB) может подаваться через преобразователь напряжения AC/DC (блок питания) или аккумуляторной батареей. Преобразователь напряжения подключается посредством разъема 2.1 мм с центральным положительным полюсом. Провода от батареи подключаются к выводам Gnd и Vin разъема питания.

Основная необходимая часть контроллера для работы с гироскопами необходимы контакты с возможностью работы с шиной I2C. На плате отдельно выведены контакты SDA и SCL как раз для шины I2C. На данной плате это выводы A4 и A5. В других версиях платы данные выводы могут быть выведены на другие пины.

**1.2 Архитектура микроконтроллеров AVR.**

Основой микроконтроллеров AVR является 8–битное микропроцессорное ядро или центральное процессорное устройство (ЦПУ), построенное на принципах RISC–архитектуры. Основой этого блока служит арифметико–логическое устройство (АЛУ). По системному тактовому сигналу из памяти программ в соответствии с содержимым счетчика команд (Program Counter – PC) выбирается очередная команда и выполняется АЛУ. Во время выбора команды из памяти программ происходит выполнение предыдущей выбранной команды, что и позволяет достичь быстродействия 1 MIPS на 1 МГц. Архитектура микроконтроллера AVR представлена на рисунке 2.

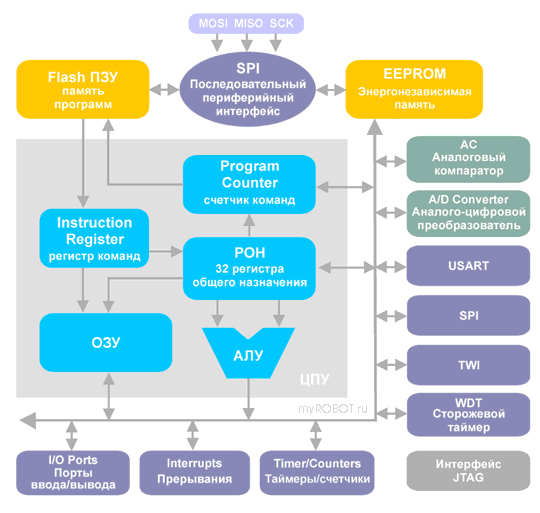


Рисунок 2 – Архитектура микроконтроллера AVR

АЛУ подключено к регистрам общего назначения РОН (General Purpose Registers – GPR). Регистров общего назначения всего 32, они имеют байтовый формат, то есть каждый из них состоит из восьми бит. РОН находятся в начале адресного пространства оперативной памяти, но физически не являются ее частью. Поэтому к ним можно обращаться двумя способами (как к регистрам и как к памяти). Такое решение является особенностью AVR и повышает эффективность работы и производительность микроконтроллера. Отличие между регистрами и оперативной памятью состоит в том, что с регистрами можно производить любые операции (арифметические, логические, битовые), а в оперативную память можно лишь записывать данные из регистров.

## **1.2 Node MCU v.3 на базе ESP8266.**

NodeMCU – это платформа IoT (интернет вещей) с открытым исходным кодом. Он включает в себя встроенное программное обеспечение, которое работает на ESP8266 Wi–Fi SoC (однокристальная система) от Espressif Systems, и аппаратное обеспечение, основанное на модуле ESP–12. Термин «NodeMCU» относится не к комплектам для разработки, а к прошивке. В прошивке используется язык сценариев Lua. Он был основан на проекте eLua и основан на Espressif Non–OS SDK для ESP8266.

NodeMCU был создан вскоре после выхода ESP8266. 30 декабря 2013 года Espressif Systems начала производство ESP8266. ESP8266 – это Wi–Fi SoC, интегрированный с ядром Tensilica Xtensa LX106, широко используемое в приложениях IoT. NodeMCU был запущен 13 октября 2014 года, когда Хонг передал первый файл nodemcu–firmware на GitHub. Два месяца спустя, проект расширился и включил открытую аппаратную платформу, когда разработчик Хуанг Р. передал файл gerber платы ESP8266 с именем devkit v0.9. Позже в том же месяце Tuan PM перенес клиентскую библиотеку MQTT из Contiki на платформу SoC ESP8266 и посвятил себя проекту NodeMCU, затем NodeMCU смог поддержать протокол IQT MQTT, используя Lua для доступа к брокеру MQTT. Еще одно важное обновление было сделано 30 января 2015 года, когда Devsaurus портировал u8glib в проект NodeMCU, позволяя NodeMCU легко управлять ЖК–дисплеями, экранами, OLED и даже VGA–дисплеями. Летом 2015 года создатели отказались от проекта прошивки, и группа независимых разработчиков вступила во владение. К лету 2016 года в NodeMCU вошло более 40 различных модулей. Из–за нехватки ресурсов пользователям необходимо выбрать модули, соответствующие их проекту, и создать прошивку с учетом их потребностей.

ESP8266 может работать как в роли точки доступа, так и конечной станции. При нормальной работе в локальной сети ESP8266 конфигурируется в режим конечной станции. Для этого устройства необходимо задать SSID Wi–Fi сети и в закрытых сетях, пароль доступа. Для первоначального конфигурирования этих параметров удобен режим точки доступа. В режиме точки доступа устройство видно при стандартном поиске сетей в планшетах и компьютерах. Остается подключиться к устройству, открыть HTML страничку конфигурирования и задать сетевые параметры. После чего устройство штатно подключится к локальной сети в режиме конечной станции. В случае исключительно местного использования возможно всегда оставлять устройство в режиме точки доступа, что снижает необходимые усилия пользователя по его настройке.

## **1.3 Платформа Node JS на языке Java Script**

JavaScript – мультипарадигменный язык программирования. Поддерживает объектно–ориентированный, императивный и функциональный стили. Является реализацией языка ECMAScript (стандарт ECMA–262).

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб–страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса. На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java, но при этом лёгким для использования непрограммистами. Языком JavaScript не владеет какая–либо компания или организация, что отличает его от ряда языков программирования, используемых в веб–разработке.

**1.3.1 Семантика и синтаксис**

Синтаксис языка JavaScript во многом напоминает синтаксис Си и Java, семантически же язык гораздо ближе к Self, Smalltalk или даже Лиспу.

В JavaScript:

Все идентификаторы регистрации:

* в названиях переменных можно использовать буквы, подчёркивание, символ доллара, арабские цифры;
* названия переменных не могут начинаться с цифры;
* для оформления однострочных комментариев используются //, многострочные и внутристрочные комментарии начинаются с /\* и заканчиваются.
* Node или Node.js — программная платформа, основанная на движке V8 (трансчсммлирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения.
* Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода–вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript–кода.

Node.js применяется преимущественно на сервере, выполняя роль web–сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и macOS) и даже программировать микроконтроллеры (например, tessel и espruino). В основе Node.js лежит событийно–ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

Создание и запуск HTTP–сервера на Node.js, выдающего Hello, world!:

// Загружаем модуль http

const http = require('http');

// Создаем web–сервер с обработчиком запросов

const server = http.createServer((req, res) => {

console.log('Начало обработки запроса');

// Передаем код ответа и http–заголовки

res.writeHead(200, {

'Content–Type': 'text/plain; charset=UTF–8'

});

res.end('Hello world!');

});

// Запускаем web–сервер

server.listen(3000, '127.0.0.1', () => {

console.log('Сервер запущен http://127.0.0.1:3000/');

});

## **1.4 Преимущества Node.js перед другими платформами для Web–программирования**

Передовой и крайне интересной платформой для реализации веб–приложений, является Node.js. Node.js сравнительно молодой проект: разработчики узнали о выходе идейно новой технологии разработки веб–ресурсов в 2009 году. С тех пор платформа разрослась и достаточно комфортно используется многими программистами по всему миру. Чем определена популярность Node.js? Первое и самое полезное свойство платформы, особенно для новичков – простота и доступность. Для реализации собственных проектов вам не придется вручную по крупицам собирать библиотеки, пакеты, информацию о данном продукте. Установка занимает считанные минуты, после которых Node уже готов к работе.

Структурно Node.js представляет собой объединение движка ([V8](https://ru.wikipedia.org/wiki/V8_(%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA_JavaScript))), средств ввода–вывода(I/O) и набора библиотек. При этом разработка компонентов приложения или сайта ведется на одном языке программирования – JavaScript. Не требуется изучение дополнительных ресурсов или привлечение специалиста другого языка, достаточно обладать хорошими навыками программирования на JS, изучение которого, в свою очередь, не затруднительно.

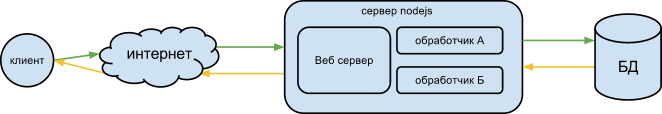


Рисунок 3 – Модель работы сервера на Node.js.

Node исповедует событийно–ориентированнуюпарадигму программирования в соответствии с которой события определяют выполнение программы, построение приложения происходит с помощью программирования обработчиков множества событии, которые могут произойти в процессе выполнения.

Еще одно преимущество данной платформы – разработана для всех типов ОС. Установка Node.js одинаково проста и на Mac, и на Windows, и на Unix системах. Тем самым укрепляется доступность и стабильность данного продукта.

Углубимся в суть платформы. Ранее был упомянут движок V8. На данный момент это одна из самых производительных программ для JS. Благодаря ей код выполняется в разы быстрее. V8 эффективно управляет памятью:

* Прерывания для сборки мусора;
* Минимизация влияния этих прерываний на время выполнения кода;
* Оптимальное хранение указателей и объектов, как следствие – ликвидация проблемы потери памяти;
* Быстрый доступ к информации;

Тандем из V8 и набора библиотек Node.js реализует асинхронное API по работе с сетевыми ресурсами. Теперь сервер не подвисает в ожидании ответа для запроса определенного процесса, например, от базы данных, а продолжает работу с другими запросами, пока данный находится в процессе выполнения. Это достаточно большой шаг для интерактивных систем, где важна скорость ответа на определенные действия на клиенте. Схема работы сервера Node.js представлена на рисунке 4.

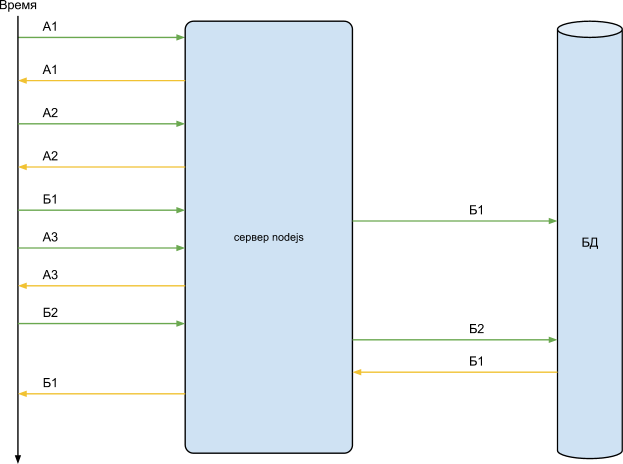


Рисунок 4 – схема работы сервера Nodejs.

Итог:

* Легко работать;
* Прототип;
* Интерактивная разработка;
* Легко выстраивать архитектуру (готовые пакеты);

## **1.5 Mongo DB**

MongoDB— документно–ориентированная система управления базами данных (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON–подобны Система поддерживает ad–hoc–запросы: они могут возвращать конкретные поля документов и пользовательские JavaScript–функции. Поддерживается поиск по регулярным выражениям. Также можно настроить запрос на возвращение случайного набора результатов. Имеется поддержка индексов. Система может работать с набором реплик, то есть, содержать две или более копии данных на различных узлах. Каждый экземпляр набора реплик может в любой момент выступать в роли основной или вспомогательной реплики. Все операции записи и чтения по умолчанию осуществляются с основной репликой. Вспомогательные реплики поддерживают в актуальном состоянии копии данных. В случае, когда основная реплика дает сбой, набор реплик проводит выбор, который из реплик должен стать основным. Второстепенные реплики могут дополнительно являться источником для операций чтения. Система масштабируется горизонтально, используя технику сегментирования объектов баз данных — распределение их частей по различным узлам кластера. Администратор выбирает ключ сегментирования, который определяет, по какому критерию данные будут разнесены по узлам (в зависимости от значений хэша ключа сегментирования). Благодаря тому, что каждый узел кластера может принимать запросы, обеспечивается балансировка нагрузки. Система может быть использована в качестве файлового хранилища с балансировкой нагрузки и репликацией данных. Предоставляются программные средства для работы с файлами и их содержимым. GridFS используется в плагинах для Nginx и lighttpd. GridFS разделяет файл на части и хранит каждую часть как отдельный документ. Может работать в соответствии с парадигмой MapReduce. В фреймворке для агрегации есть аналог SQL–инструкции GROUP BY. Операторы агрегации могут быть связаны в конвейер подобно UNIX–конвейрам. Фреймворк так же имеет оператор $lookup для связки документов при выгрузке и статистические операции такие как среднеквадратическое отклонение. Поддерживается JavaScript в запросах, функциях агрегации (например, в MapReduce).

MongoDB поддерживает коллекции с фиксированным размером. Такие коллекции сохраняют порядок вставки и по достижении заданного размера ведут себя как кольцевой буфер.

В июне 2018 года (в версии 4.0) добавлена поддержка транзакций, удовлетворяющих требованиям ACID документы и схему базы данных. Написана на языке C++.

## **1.6 Преимущества Mongo DB**

* Основные преимущества MongoDB:
* Отсутствие схемы
* Данная БД основана на коллекциях различных документов. Количество полей, содержание и размер этих документов может отличаться.
* Крайне понятная структура каждого объекта.
* Легко масштабируется
* Для хранения используемых в данный момент данных используется внутренняя память, что позволяет получать более быстрый доступ.
* Данные хранятся в виде JSON документов
* MongoDB поддерживает динамические запросы документов (document–based query)
* Отсутствие сложных JOIN запросов
* Нет необходимости маппинга объектов приложения в объекты БД

Подведя итого можно сказать, что MongoDB является достаточно неплохим решением, если мы имеем дело с Big Data.

На этом мы заканчиваем обзор преимуществ MongoDB.

**1.7 Проблемы которые будет решать наш Web–интерфейс.**

Данный Web–интерфейс кроме того, что служит отображением данных получаемых с датчиков, также решает такие проблемы, как упрощение документооборота для медицинских учреждений, путем конвертирования карточек пациентов в электронный формат, что облегчает доступ к ним медработников, а также выстраивает полную картину лечения пациента, с помощью отображения графиков его биологических показателей. Также он сильно облегчает работу с устройством путем привязывания. Устройства напрямую к карточке пациента.

# 2 **Разработка серверной части системы мониторинга медицинских показателей**

**2.1 Разработка web– интерфейса**

На Рисунке 5 представлена страница авторизации – это окно, с помощью которого медицинские работники. Она содержит в себе кнопку «Войти», цвет «серый», два поля для ввода.



Рисунок 5 – Форма авторизации на Веб–интерфейс

На рисунке 6 представлена страница регистрации – это окно, с помощью которого медицинские работники смогут зарегистрироваться. Она содержит в себе кнопку «Создать», цвет «серый», четыре поля для ввода.

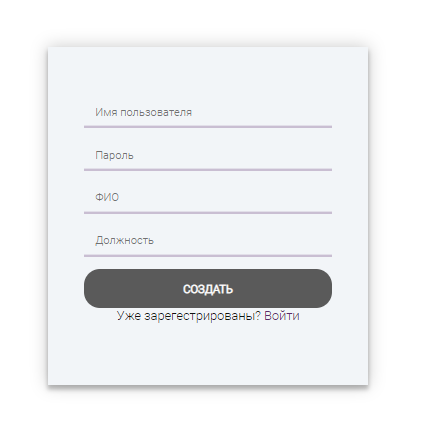


Рисунок 6 – Форма регистрации нового пользователя

На рисунке 7 представлена главная страница – это информация, которую, в большинстве случаев, пользователь видит первой при переходе на сайт, именно поэтому на ней отображается основная информация о сайте.



Рисунок 7 – Главная страница

Шапка сайта – это титульная верхняя часть дизайн–макета веб–страницы. Основное назначение этого элемента – дать четкое обозначение начала страницы.

Шапка сайта является важной составляющей макета и представляет собой сквозной элемент (общий для всех страниц сайта), на котором располагают такую основную информацию, как логотип, меню навигации и другой важный контент.

Логотип находится в левом верхнем углу. Меню навигации состоит из ссылок: «Карточки пациентов», «Работники», «Выйти». Основной цвет ссылок #5A2A6A.

Слайдер содержит 3 изображений с надписью. Изображения имеют разрешение 2843х1962.

На рисунке 8 представлена страница с карточками пациентов – страница с общими данными о больных.

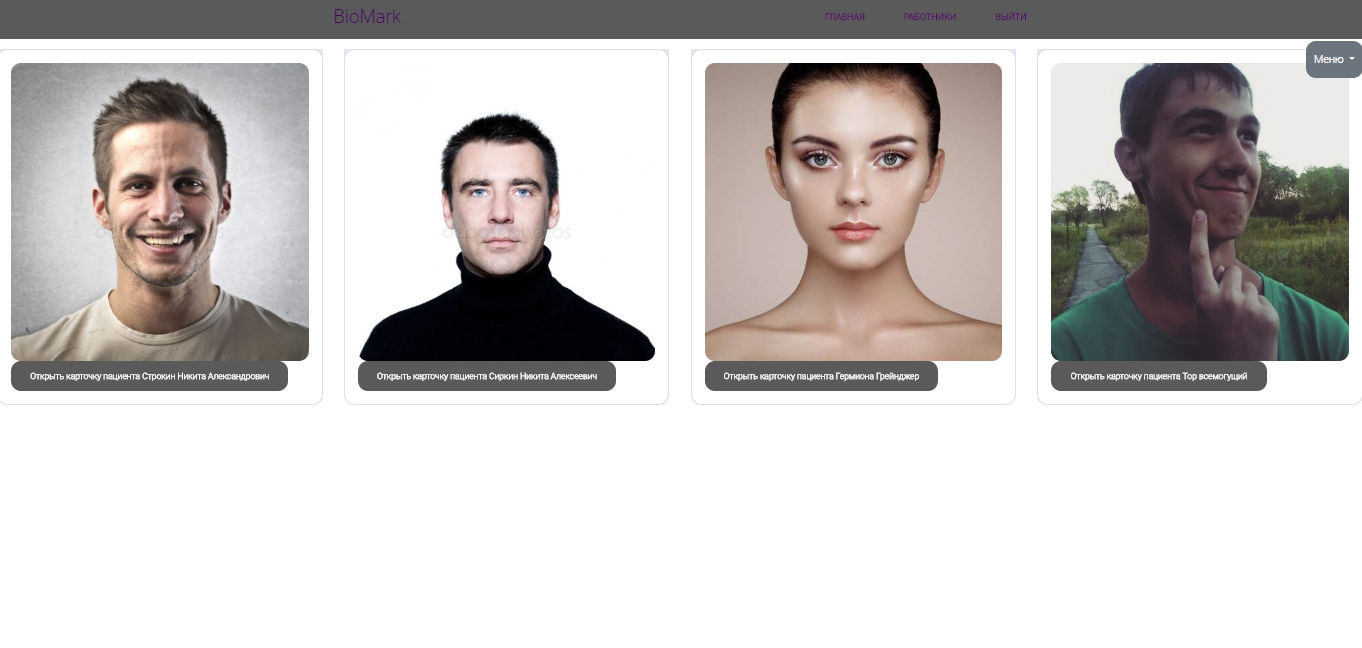


Рисунок 8– Карточки пациентов

У страницы есть заголовок: «Карточки пациентов». Цвет шрифта #5A2A6A, цвет фона #AAAAA, позиционирование – фиксированное.

На рисунке 9 представлена страница Форма карточки пациента – страница с подробными данными больного и его температура (см. Рисунок 9).

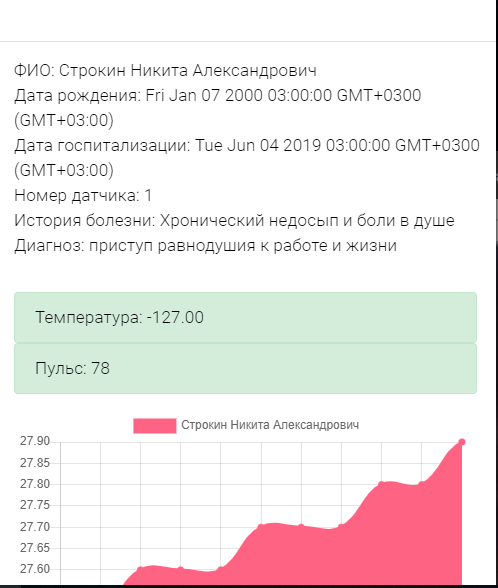


Рисунок 9 – Форма карточки пациента

Цвет шрифта «white», цвет фона #FFFFFF, позиционирование – фиксированное.

На рисунке 10 представлена страница работников – на ней показана общая информация работников. Заголовок: «Работники». Цвет шрифта «white», цвет фона #FFFFFF, позиционирование – фиксированное.



Рисунок 10 – Страница «Работники»

У формы есть заголовок: «Форма подробной информации о работнике». Цвет шрифта «white», цвет фона #AAAAA, позиционирование – фиксированное.

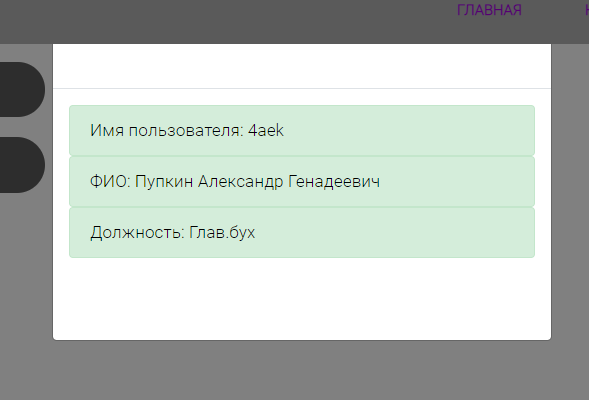


Рисунок 11 – Подробная информация о работнике

На этой форме отображаются подробные данные работника.

## **2.5 Структура базы данных**

На данном Веб интерфейсе была реализована база данных через mongoLab.

mLab – это полностью управляемая облачная служба баз данных, в которой размещаются базы данных MongoDB. mLab работает с облачными провайдерами Amazon, Google и Microsoft Azure и сотрудничает с поставщиками платформ как услуг. В мае 2011 года компания mLab получила финансирование в первом раунде на сумму 3 млн. На рисунке 11 представлена база данных для Web – сайта на интернет – сервисе mongoLab.

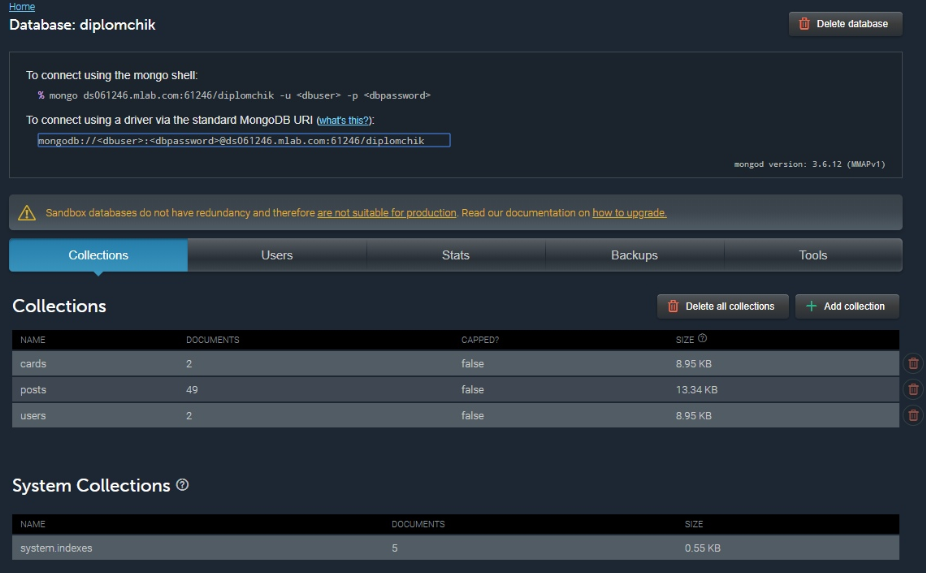


Рисунок 12 – Интерфейс mLab

**2.6 Служба поддержки**

Nodemailer – это модуль для приложений Node.js, позволяющий легко отправлять электронные письма. Проект начался в 2010 году, когда не было никакой разумной возможности отправлять электронные письма, сегодня это решение, к которому большинство пользователей Node.js обращаются по умолчанию.

Особенности Nodemailer:

* Единый модуль с нулевыми зависимостями – код легко проверяется, так как нет темных углов;
* Сосредоточение внимания на безопасности, никто не любит [уязвимости RCE](https://thehackernews.com/2017/01/phpmailer-swiftmailer-zendmail.html);
* Поддержка Юникода для использования любых персонажей, включая эмодзи;

Поддержка Windows – вы можете установить его с помощью [npm](https://www.npmjs.com/package/nodemailer) в Windows, как и любой другой модуль, скомпилированные зависимости отсутствуют. Используйте это без проблем из Azure или из вашего окна Windows;

* Использовать HTML–контент, а также текстовую альтернативу
* Добавить [вложения](https://nodemailer.com/message/attachments/) в сообщения;
* [Встроенные](https://nodemailer.com/message/embedded-images/) вложения изображений для HTML–контента – ваш дизайн не блокируется;
* Безопасная доставка электронной почты с использованием TLS / STARTTLS;
* Различные [способы транспортировки](https://nodemailer.com/transports/) в дополнение к встроенной [поддержке SMTP](https://nodemailer.com/smtp/);
* Подписывать сообщения с [DKIM](https://nodemailer.com/dkim/);
* [Поддержка](https://nodemailer.com/plugins/) пользовательских [плагинов](https://nodemailer.com/plugins/) для управления сообщениями;
* [Прокси](https://nodemailer.com/smtp/proxies/) для SMTP–соединений;
* Автоматически сгенерированные почтовые тестовые аккаунты от [Ethereal.email](https://ethereal.email/).

**ЗАКЛЮЧЕНИЕ**

За время выполнения выпускной квалификационной работы был разработан сервер и web–сайт для устройства мониторинга медицинских показателей «Biomark».

В процессе разработки ставилась задача спроектировать сервер и web – сайт для устройства мониторинга, который отличался простым и удобным интерфейсом. Цели и задачи выполнены в полном объёме. Были использованы различные интернет – сервисы. Также было проведено тестирование сервера и web – cайта.

Были выполнены проектирование, разработка, программирование и отладка устройства. В процессе разработки, также были решены следующие проблемы:

* в связи с задержкой в скорости передачи данных, с устройства на сервер, для адекватной работы потребовалось провести оптимизацию кода посредством переосмысления используемых типов данных, а также использования функций;
* возникла проблема в поиске нужного языка программирования для написания сервера;
* Разработанный сервер и web–сайт обеспечивает оперативный доступ к медицинским показателям человека.

Достоинства устройства:

* практичный и удобный web – интерфейс для взаимодействия с сервером;
* web – сайт работает на всех браузерах;
* используется алгоритм, экономящий интернет–трафик и время загрузки страниц;

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Алексеев А.П. Введение в web дизайн [Электронный ресурс] / Алексеев А.П. — Электрон. дан. — Москва : Издательство "СОЛОН–Пресс", 2008. — 192 с. — Режим доступа: https://e.lanbook.com/book/13768?category=1538
2. Адамс Д.Р., Флойд К.С. Основы работы с XHTML и CSS [Электронный ресурс] / Адамс Д.Р., Флойд К.С. — Электрон. дан. — Москва : Национальный Открытый Университет "ИНТУИТ", 2016. — 567 с. — Режим доступа: https://e.lanbook.com/book/100331?category=1556
3. Введение в стандарты Web [Электронный ресурс] : учебное пособие. — Электрон. дан. — Москва : , 2016. — 800 с. — Режим доступа: https://e.lanbook.com/book/100712.
4. Кириченко, А.В. HTMLS + CSS3. Основы современного WEB–дизайна [Электронный ресурс] : руководство / А.В. Кириченко, А.А. Хрусталев. — Электрон. дан. — Санкт–Петербург : Наука и Техника, 2018. — 352 с. — Режим доступа: https://e.lanbook.com/book/108277.
5. Основы работы с CSS [Электронный ресурс] : учебное пособие. — Электрон. дан. — Москва : , 2016. — 195 с. — Режим доступа: https://e.lanbook.com/book/100327.
6. Основы работы с Photoshop [Электронный ресурс] / учебное пособие— Электрон. дан. — Москва : Национальный Открытый Университет "ИНТУИТ", 2016. — 1393 с. — Режим доступа: https://e.lanbook.com/book/100338?category=1538
7. Панфилов К. Создание веб сайта от замысла до реализации [Электронный ресурс] / Панфилов К. — Электрон. дан. — Москва : Издательство "ДМК Пресс", 2009. — 440 с. — Режим доступа: https://e.lanbook.com/book/1072?category=1538
8. Романычева Э.Т., Яцюк О.Г. Дизайн и реклама. Компьютерные технологии: Справочное и практическое руководство [Электронный ресурс] / Романычева Э.Т., Яцюк О.Г. — Электрон. дан. — Москва : Издательство "ДМК Пресс", 2006. — 432 с. — Режим доступа: https://e.lanbook.com/book/1102?category=1538
9. Сухов К. HTML5 – путеводитель по технологии [Электронный ресурс] / Сухов К. — Электрон. дан. — Москва : Издательство "ДМК Пресс", 2013. — 352 с. — Режим доступа: https://e.lanbook.com/book/40002?category=1538
10. Сычев А.В. Web – технологии [Электронный ресурс] / Сычев А.В. — Электрон. дан. — Москва : Национальный Открытый Университет "ИНТУИТ", 2016. — 408 с. — Режим доступа: https://e.lanbook.com/book/100725?category=1538

**ЗАКЛЮЧЕНИЕ**

За время выполнения выпускной квалификационной работы был спроектирован разработан сайт для устройства мониторинга медицинских показателей «Biomark».

В процессе разработки ставилась задача спроектировать сервер и web – сайт для устройства мониторинга, который отличался простым и удобным интерфейсом. Цели и задачи выполнены в полном объёме. Были использованы различные интернет – сервисы. Также было проведено тестирование сервера и web – cайта.

Были выполнены проектирование, разработка, программирование и отладка устройства. В процессе разработки, также были решены следующие проблемы:

* И связи с задержкой в скорости передачи данных, с устройства на сервер, для адекватной работы потребовалось провести оптимизацию кода посредством переосмысления используемых типов данных, а также использования функций;
* Возникла проблема в поиске нужного языка программирования для написания сервера;
* Разработанный сервер и web–сайт обеспечивает оперативный доступ к медицинским показателям человека.
* Достоинства устройства:
* Практичный и удобный интерфейс для взаимодействия с сервером;
* Web – сайт работает на всех браузерах;
* Используется алгоритм, экономящий интернет–трафик и время загрузки страниц;
* Разработанное устройство обеспечивает оперативный доступ к медицинским показателям человека.
* практичность и удобство в эксплуатации сайта;
* мониторинг данных ведется через любое устройство с доступом в интернет;
* используется алгоритм, экономящий интернет–трафик;
* адаптивный дизайн;

**ПРИЛОЖЕНИЕ А**

Техническое задание

**Наименование программы**

Системы мониторинга медицинских показателей.

**Краткая характеристика области применения**

Данный сайт применяется в медицинских учреждениях.

**Основание для разработки**

Разработка проводится на основании задания на дипломный проект, для студентов Института среднего профессионального образования (ИНСПО), обучающихся по специальности 09.03.02 «Программирование в компьютерных системах».

**Наименование темы разработки**

«Разработка серверной части системы мониторинга медицинских показателей».

**Руководителем разработки является** – Титов Н.Г.

**Назначение разработки**

Web – cайт предназначен для устройства мониторинга медицинских показателей. Является интерфейсом для устройства «Biomark».

**Требования по информационному обеспечению сайта**

* сбор необходимой информации, ее систематизация и подготовка для информационного обеспечения сайта;
* обработка, редактирование и вычитка информационных материалов (текстов, изображений, каталогов, описаний и так далее) для информационного обеспечения сайта;
* размещение на сайте специальных предложений, информации по различным акциям;
* проверка актуальности материалов.

**Требования к защите данных от разрушений**

* Сбои и аварии не должны приводить к необратимой потере информации серверов и хранилищ данных.
* Для обеспечения сохранности информации при аварийных ситуациях в ИС УОС должны быть предусмотрены средства обеспечения бесперебойного питания, дублирования информации.

**Требования к контролю, обновлению и восстановлению данных**

* Данные должны обновляться по мере устаревания или потери актуальности вручную каждую неделю. Должно быть предусмотрено резервное копирование (архивирование).
* Контроль, хранение, восстановление данных должны быть регламентированы. Хранение резервных данных должно осуществляться в местах, не допускающих их повреждения, в случае уничтожения или повреждения основных данных ИС УОС.
* Система должна обеспечивать восстановление данных в случае разрушения аппаратного комплекса при авариях и сбоях в электропитании системы средствами системы управления базой данных, аппаратными средствами серверного оборудования, прикладными средствами резервного копирования.

**ПРИЛОЖЕНИЕ Б**

Листининг программы

'use strict';

var config = require('../config');

var passport = require('passport');

var logger = require('../logger');

var LocalStrategy = require('passport–local').Strategy;

var User = require('../models/user');

var init = function(){

passport.serializeUser(function(user, done) {

done(null, user.id);

});

passport.deserializeUser(function(id, done) {

User.findById(id, function (err, user) {

done(err, user);

});

});

passport.use(new LocalStrategy(

function(username, password, done) {

User.findOne({ username: new RegExp(username, 'i')}, function(err, user) {

if (err) { return done(err); }

if (!user) {

return done(null, false, { message: 'Неправильное имя пользователя или пароль.' });

}

user.validatePassword(password, function(err, isMatch) {

if (err) { return done(err); }

if (!isMatch){

return done(null, false, { message: 'Неправильное имя пользователя или пароль.' });

}

return done(null, user);

});

});

}

));

return passport;

}

module.exports = init();

{

"sessionSecret": "asfjaip",

"redis": {

"host": "127.0.0.1",

"port": 6379,

"password": ""

}

}

––––––––––––––––

'use strict';

var init = function () {

if(process.env.NODE\_ENV === 'production') {

var redisURI = require('url').parse(process.env.REDIS\_URL);

var redisPassword = redisURI.auth.split(':')[1];

return {

sessionSecret: process.env.sessionSecret,

redis: {

host: redisURI.hostname,

port: redisURI.port,

password: redisPassword

}

}

}

else {

return require('./config.json');

}

}

module.exports = init();

––––––––––––––––––––––––––––––

'use strict';

var config = require('../config');

var Mongoose = require('mongoose');

var logger = require('../logger');

Mongoose.set('useFindAndModify', false);

var dbURI = "mongodb://test:testtest1@ds061246.mlab.com:61246/diplomchik";

Mongoose.connect(dbURI, { useNewUrlParser: true });

Mongoose.connection.on('error', function(err) {

if(err) throw err;

});

Mongoose.Promise = global.Promise;

module.exports = { Mongoose,

models: {

user: require('./schemas/user.js'),

card: require('./schemas/card.js'),

post: require('./schemas/post.js')

}

};

–––––––––––

'use strict';

var Mongoose = require('mongoose');

var CardSchema = new Mongoose.Schema({

FIO: { type: String, required: true},

Datebirthday: { type: Date, default: null },

Dategospital: { type: Date, default: null },

Datchik: { type: Number, required: true},

History: { type: String, default: null},

Diagnoz: {type: String, default: null}

});

CardSchema.pre('save', function(next) {

var user = this;

next();

});

–––––––––––––

'use strict';

var Mongoose = require('mongoose');

var PostSchema = new Mongoose.Schema({

ids: { type: Number, },

temp1: { type: String, default: null },

pulse: { type: Number, default: null },

timer: { type: Number, default: null }

});

var postModel = Mongoose.model('post', PostSchema);

module.exports = postModel;

–––––––––––––––––––––––––

'use strict';

var Mongoose = require('mongoose');

var bcrypt = require('bcrypt–nodejs');

const SALT\_WORK\_FACTOR = 12;

const DEFAULT\_USER\_PICTURE = "https://st2.depositphotos.com/5266903/8486/v/450/depositphotos\_84867024–stock–illustration–surgeon–icon.jpg";

var UserSchema = new Mongoose.Schema({

username: { type: String, required: true},

password: { type: String, default: null },

picture: { type: String, default: DEFAULT\_USER\_PICTURE},

FIO: { type: String, default: null},

Rang: { type: String, default: null},

});

UserSchema.pre('save', function(next) {

var user = this;

if(!user.picture){

user.picture = DEFAULT\_USER\_PICTURE;

}

if (!user.isModified('password')) return next();

bcrypt.genSalt(SALT\_WORK\_FACTOR, function(err, salt) {

if (err) return next(err);

bcrypt.hash(user.password, salt, null, function(err, hash) {

if (err) return next(err);

user.password = hash;

next();

});

});

});

UserSchema.methods.validatePassword = function(password, callback) {

bcrypt.compare(password, this.password, function(err, isMatch) {

if (err) return callback(err);

callback(null, isMatch);

});

};

var userModel = Mongoose.model('user', UserSchema);

module.exports = userModel;

––––––––––––––––––––––––––––––––––––––––––––––––––

'use strict';

var winston = require('winston');

var logger = new (winston.Logger)({

transports: [

new (winston.transports.File)({

level: 'debug',

json: true,

filename: './debug.log',

handleExceptions: true

}),

new (winston.transports.Console)({

level: 'debug',

json: true,

handleExceptions: true

})

],

exitOnError: false

});

module.exports = logger;

–––––––––––––

'use strict';

const nodemailer = require('nodemailer');

const xoauth2 = require('xoauth2');

var transporter = nodemailer.createTransport({

service: 'gmail',

auth : {

xoauth2: xoauth2.createXOAuth2Generator({

user : 'coderesponse200@gmail.com',

clientId: '390771420202–l5jc4tkcdno1a2a05puk4pfcg9t16s4c.apps.googleusercontent.com',

clientSecret: 'rcdkNKTXrO\_q3GWViorhkOee',

refreshToken: '1/XN3VDokXeU1MFhyRVVbsmJzWy5FgUbmFMKAOKkqRpes'

})

}

})

var mailOptions = {

from: 'Tech–squad <coderesponse200@gmail.com>',

to: 'coderesponse200@gmail.com',

subject: 'nodemailer test',

text: 'Hello bitch'

}

transporter.sendMail(mailOptions, function(err, res) {

if(err){

console.log('Error');

} else {

console.log('Email sent');

}

})

–––––––––

'use strict';

var cardModel = require('../database').models.card;

var create = function (data, callback){

var newCard = new cardModel(data);

newCard.save(callback);

};

var findOne = function (data, callback){

cardModel.findOne(data, callback);

}

var findByDatchik = function (Datchik, callback){

cardModel.findByDatchik(Datchik, callback);

}

var findByFIOAndUpdate = function(FIO, data, callback){

cardModel.findByFIOAndUpdate(FIO, data, { new: true }, callback);

}

module.exports = {

create,

findOne,

findByDatchik,

findByFIOAndUpdate

};

––––––––––––––––––––––––––––

'use strict';

var postModel = require('../database').models.post;

var create = function (data, callback){

var newPost = new postModel(data);

newPost.save(callback);

};

var findOne = function (data, callback){

postModel.findOne(data, callback);

}

var findById = function (id, callback){

postModel.findById(id, callback);

}

var findOneAndUpdate = function(timer, data, callback){

postModel.findOneAndUpdate(timer, data, { new: true }, callback);

}

var find = function (data, callback){

postModel.find(data, callback);

}

module.exports = {

create,

findOne,

findById,

findOneAndUpdate,

find

};

–––––––––––––––––––––––––––––––––––––

'use strict';

var userModel = require('../database').models.user;

var create = function (data, callback){

var newUser = new userModel(data);

newUser.save(callback);

};

var findOne = function (data, callback){

userModel.findOne(data, callback);

}

var findById = function (id, callback){

userModel.findById(id, callback);

}

var findOrCreate = function(data, callback){

findOne({'socialId': data.id}, function(err, user){

if(err) { return callback(err); }

if(user){

return callback(err, user);

} else {

var userData = {

username: data.displayName,

socialId: data.id,

picture: data.photos[0].value || null

};

if(data.provider == "facebook" && userData.picture){

userData.picture = "http://graph.facebook.com/" + data.id + "/picture?type=large";

}

create(userData, function(err, newUser){

callback(err, newUser);

});

}

});

}

var isAuthenticated = function (req, res, next) {

if(req.isAuthenticated()){

next();

}else{

res.redirect('/');

}

}

module.exports = {

create,

findOne,

findById,

findOrCreate,

isAuthenticated

};

'use strict';

var express = require('express');

var router = express.Router();

var passport = require('passport');

var mongo = require('mongodb').MongoClient;

var url = 'mongodb://test:testtest1@ds061246.mlab.com:61246/diplomchik';

var User = require('../models/user');

var Card = require('../models/card');

var Post = require('../models/post');

var assert = require('assert');

router.get('/', function(req, res, next) {

if(req.isAuthenticated()){

res.redirect('/main');

}

else{

res.render('login', {

success: req.flash('success')[0],

errors: req.flash('error'),

showRegisterForm: req.flash('showRegisterForm')[0]

});

}

});

router.post('/login', passport.authenticate('local', {

successRedirect: '/main',

failureRedirect: '/',

failureFlash: true

}));

router.post('/register', function(req, res, next) {

var credentials = {'username': req.body.username, 'password': req.body.password, 'FIO' : req.body.FIO, 'Rang' : req.body.Rang };

if(credentials.username === '' || credentials.password === '' || credentials.FIO === ''){

req.flash('error', 'Вы что–то не заполнили');

req.flash('showRegisterForm', true);

res.redirect('/');

}else{

User.findOne({'username': new RegExp('^' + req.body.username + '$', 'i')}, function(err, user){

if(err) throw err;

if(user){

req.flash('error', 'Имя пользователя уже занято.');

req.flash('showRegisterForm', true);

res.redirect('/');

}else{

User.create(credentials, function(err, newUser){

if(err) throw err;

req.flash('success', 'Ваш аккаунт создан. Пожалуйста войдите.');

res.redirect('/');

});

}

});

}

});

––––––––––––

var timer = 0;

var timerCallback = function() {

if (timer<48){

timer = timer + 1;

}else{

timer = 0;

};

};

router.get('/temp', function(req, res) {

console.log(req.query);

var timerId = setInterval(timerCallback.bind(this), 1800000);

var dat = {'temp1': req.query.temp1, 'pulse': req.query.pulse , 'ids': req.query.ids, 'timer' : timer };

Post.findOne({ 'ids' : req.query.ids}, function(err, havePost){

if(err) throw err;

if(havePost){

Post.findOneAndUpdate({'timer' : timer}, {$set:{ temp1: req.query.temp1, pulse : req.query.pulse}},function(err, posto){

if(err) throw err;

res.sendStatus(200);

});

}else{

Post.create(dat, function(err, newPosts){

if(err) throw err;

res.sendStatus(200);

});

}

});

});

––––––––––––––

router.get('/card', [User.isAuthenticated, function(req, res, next) {

var resultArray = [];

var resultDat =[];

var resultArray2 = [];

var resultDat2 =[];

mongo.connect(url, function(err, db) {

assert.equal(null, err);

var cursor = db.collection('cards').find({ Datchik : 1 });

assert.equal(null, err);

var Dathiki = db.collection('posts').find({ ids : 1 , timer : timer});

assert.equal(null, err);

var cursor2 = db.collection('cards').find({ Datchik : 2 });

assert.equal(null, err);

var Dathiki2 = db.collection('posts').find({ ids : 2 });

cursor.forEach(function(doc, err) {

assert.equal(null, err);

resultArray.push(doc);

});

––––––––––––––––––

cursor2.forEach(function(doc, err) {

assert.equal(null, err);

resultArray2.push(doc);

});

Dathiki.forEach(function(doc, err) {

assert.equal(null, err);

resultDat.push(doc);

});

––––––––––––––––––––––––

Dathiki2.forEach (function (doc, err) {

assert.equal(null, err);

resultDat2.push(doc);

}, function () {

db.close();

res.render('card.hbs', {good: resultDat, items: resultArray, good2: resultDat2, items2: resultArray2 });

});

});

}]);

router.post('/card', [User.isAuthenticated, function(req, res) {

if (!req.body) return res.sendStatus(400);

console.log (req.body);

var dani = {'FIO': req.body.FIO, 'Datebirthday': req.body.Datebirthday,

'Dategospital': req.body.Dategospital, 'Datchik': req.body.Datchik,

'History': req.body.History, 'Diagnoz': req.body.Diagnoz };

Card.findOne({'FIO': new RegExp('^' + req.body.FIO + '$', 'i')}, function(err, card){

if(err) throw err;

if(card){

res.send('Карточка с данным ФИО уже была создана');

res.redirect('/card');

}else{

Card.findOne({'Datchik': (req.body.Datchik)}, function(err, dat){

if(err) throw err;

if(dat){

res.send('Данный датчик уже занят');

res.redirect('/card');

}else{

Card.create(dani, function(err, newCard){

if(err) throw err;

res.redirect('/card');

});

}

})

}

});

}]);

router.get('/main', [User.isAuthenticated, function(req, res, next) {

res.render('main');

}]);

router.get('/worker', [User.isAuthenticated, function(req, res, next) {

var resultUsersddf = [];

var resultUsers4aek = [];

mongo.connect(url, function(err, db) {

assert.equal(null, err);

var cursorUsersddf = db.collection('users').find({ username : "ddf" });

var cursorUsers4aek = db.collection('users').find({ username : "4aek" });

assert.equal(null, err);

cursorUsers4aek.forEach(function(doc, err) {

assert.equal(null, err);

resultUsers4aek.push(doc);

});

cursorUsersddf.forEach(function(doc, err) {

assert.equal(null, err);

resultUsersddf.push(doc);

}, function() {

db.close();

res.render('worker.hbs', { us : resultUsersddf, us2 : resultUsers4aek });

});

});

}]);

router.get('/logout', function(req, res, next) {

req.logout();

req.session = null;

res.redirect('/');

});

module.exports = router;

'use strict';

var session = require('express–session');

var MongoStore = require('connect–mongo')(session);

var db = require('../database');

var config = require('../config');

var init = function () {

if(process.env.NODE\_ENV === 'production') {

return session({

secret: config.sessionSecret,

resave: false,

saveUninitialized: false,

unset: 'destroy',

store: new MongoStore({ mongooseConnection: db.Mongoose.connection })

});

} else {

return session({

secret: config.sessionSecret,

resave: false,

unset: 'destroy',

saveUninitialized: true

});

}

}

module.exports = init();

'use strict';

var config = require('../config');

var redis = require('redis').createClient;

var adapter = require('socket.io–redis');

var ioEvents = function(io) {

io.of('/rooms').on('connection', function(socket) {

});

}

var init = function(app){

var server = require('http').Server(app);

var io = require('socket.io')(server);

io.set('transports', ['websocket']);

let port = config.redis.port;

let host = config.redis.host;

let password = config.redis.password;

let pubClient = redis(port, host, { auth\_pass: password });

let subClient = redis(port, host, { auth\_pass: password, return\_buffers: true, });

io.adapter(adapter({ pubClient, subClient }));

io.use((socket, next) => {

require('../session')(socket.request, {}, next);

});

ioEvents(io);

return server;

}

module.exports = init;

=––––––––––––––––=––––––––––

<!DOCTYPE html>

<html >

<head>

<meta charset="UTF–8">

<title>Diplom</title>

<link rel="stylesheet" href="/css/style.css">

<link rel='stylesheet prefetch' href='https://maxcdn.bootstrapcdn.com/font–awesome/4.4.0/css/font–awesome.min.css'>

</head>

<body>

<div class="container not–found–error clearfix" style="text–align: center;">

<p> Мы не нашли путь который вы искали. </p>

</div>

<script src='http://cdnjs.cloudflare.com/ajax/libs/jquery/3.1.0/jquery.min.js'></script>

</body>

</html>

–––––––––––––––––––

<!DOCTYPE html>

<html >

<head>

<meta charset="UTF–8">

<title>Diplom</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384–ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">

<link rel="stylesheet" href="/css/style.css">

</head>

<body>

<div id="wrapper">

<div id="header">

<a href="/main" class="btn btn–primary" style="border: 10px; border–color: rgb(128,137,158); ">Главная страница</a>

<a href="/worker" class="btn btn–primary" style="border: 10px; border–color: rgb(128,137,158); ">Работники</a>

<a href="/logout" class="btn btn–primary" style="border: 10px; border–color: rgb(128,137,158); ">Выйти</a>

</div>

<div id="left–sidebar">

<div class="dropdown">

<button class="btn btn–secondary dropdown–toggle" type="button" id="dropdownMenu2" data–toggle="dropdown" aria–haspopup="true" aria–expanded="false" style="border: 10px; border–color: rgb(128,137,158);">

Меню

</button>

<div class="dropdown–menu" aria–labelledby="dropdownMenu2">

<button type="button" class="dropdown–item" data–toggle="modal" data–target="#exampleModal" style="border: 10px; border–color: rgb(128,137,158);">добавить пациента</button>

<button type="button" class="dropdown–item" data–toggle="modal" data–target="#exampleModal1" style="border: 10px; border–color: rgb(128,137,158); ">служба поддержки</button>

</div>

</div>

</div>

<div id="content">

<div id="inner–content">

<div class="card" style=" width: 15rem; margin–top: 2px; margin–left: 20px; margin–right: 20px; margin–bottom: 20px;">

<img src="http://abzac.org/wp–content/uploads/2013/04/nike–150x150.jpg" alt="...">

<div class="card–body">

<button type="button" class="btn btn–primary" data–toggle="modal" data–target="#exampleModal3">

Открыть карточку пациента {{# items}} {{FIO}} {{/items}}

</button>

</div>

</div>

<div class="card" style=" width: 15rem; margin–top: 20px; margin–left: 20px; margin–right: 20px; margin–bottom: 0px;">

<img src="http://abzac.org/wp–content/uploads/2013/04/nike–150x150.jpg" class="card–img–top" alt="...">

<div class="card–body">

<button type="button" class="btn btn–primary" data–toggle="modal" data–target="#exampleModal4">

Открыть карточку пациента {{# items2}} {{FIO}} {{/items2}}

</button>

</div>

</div>

</div>

</div>

<div id="footer">

<p>MAMA EBALs</p>

</div>

</div>

<div>

<div class="modal fade" id="exampleModal1" tabindex="–1" role="dialog" aria–labelledby="exampleModalLabel" aria–hidden="true">

<div class="modal–dialog" role="document">

<div class="modal–content">

<div class="modal–header">

<h5 class="modal–title" id="exampleModalLabel">Служба поддержки</h5>

<button type="button" class="close" data–dismiss="modal" aria–label="Close">

<span aria–hidden="true">&times;</span>

</button>

</div>

<div class="modal–body">

<form>

<div class="form–group">

<label for="exampleInputEmail1">Ф.И.О.</label>

<input type="text" class="form–control" a placeholder="Введите свое ФИО">

</div>

<div class="form–group">

<label for="exampleInputEmail1">Email</label>

<input type="email" class="form–control" id="exampleInputEmail1" aria–describedby="emailHelp" placeholder="Пожалуйста введите email для обратной связи">

</div>

<div>

<label for="exampleInputEmail1">Тема обращения</label>

<input type="text" class="form–control" a placeholder="Введите тему обращения">

</div>

<div>

<label for="exampleInputPassword1">Описание</label>

<input type="text" class="form–control" placeholder="Пожалуйста опишите свою проблему">

</div>

<div class="form–check">

<input type="checkbox" class="form–check–input" id="exampleCheck1">

<label class="form–check–label" for="exampleCheck1">Я соглашаюсь на обработку своих данных</label>

</div>

</form>

</div>

<div class="modal–footer">

<button type="button" class="btn btn–secondary" data–dismiss="modal">Закрыть</button>

<button type="button" class="btn btn–primary" data–dismiss="modal">Отправить</button>

</div>

</div>

</div>

</div>

<div class="modal fade" id="exampleModal4" tabindex="–1" role="dialog" aria–labelledby="exampleModalLabel" aria–hidden="true">

<div class="modal–dialog" role="document">

<div class="modal–content">

<div class="modal–header">ФИО: {{ this.FIO }}</h5>

<button type="button" class="close" data–dismiss="modal" aria–label="Close">

<span aria–hidden="true">&times;</span>

</button>

</div>

<div class="modal–body">

<section class="get">

{{# each items2 }}

<article class="items" style="">

<div style="color:Black;">ФИО: {{ this.FIO }}</div>

<div style="color:Black;">Дата рождения: {{ this.Datebirthday }}</div>

<div style="color:Black;">Дата госпитализации: {{ this.Dategospital }}</div>

<div style="color:Black;">Номер датчика: {{ this.Datchik }}</div>

<div style="color:Black;">История болезни: {{ this.History }}</div>

<div style="color:Black;">Диагноз: {{ this.Diagnoz }}</div>

<br></br>

</article>

{{/each}}

{{# each good2 }}

<article class="good" style="align–self: left;">

<div class="alert alert–success" style="color:Black; margin–top: –16px; margin–right: 3px;">Температура: {{ this.temp1 }}</div> <p> </p>

<div class="alert alert–success" style="color:Black; margin–top: –16px; margin–right: 3px;">Пульс: {{ this.pulse }}</div>

</article>

{{/each}}

</section>

</div>

<div class="modal–footer">

<button type="button" class="btn btn–secondary" data–dismiss="modal">Закрыть</button>

</div>

</div>

</div>

</div>

<div class="modal fade" id="exampleModal3" tabindex="–1" role="dialog" aria–labelledby="exampleModalLabel" aria–hidden="true">

<div class="modal–dialog" role="document">

<div class="modal–content">

<div class="modal–header">ФИО: {{ this.FIO }}</h5>

<button type="button" class="close" data–dismiss="modal" aria–label="Close">

<span aria–hidden="true">&times;</span>

</button>

</div>

<div class="modal–body">

<section class="get">

{{# each items }}

<article class="items" style="">

<div style="color:Black;">ФИО: {{ this.FIO }}</div>

<div style="color:Black;">Дата рождения: {{ this.Datebirthday }}</div>

<div style="color:Black;">Дата госпитализации: {{ this.Dategospital }}</div>

<div style="color:Black;">Номер датчика: {{ this.Datchik }}</div>

<div style="color:Black;">История болезни: {{ this.History }}</div>

<div style="color:Black;">Диагноз: {{ this.Diagnoz }}</div>

<br></br>

</article>

{{/each}}

{{# each good }}

<article class="good" style="align–self: left;">

<div class="alert alert–success" style="color:Black; margin–top: –16px; margin–right: 3px;">Температура: {{ this.temp1 }}</div> <p> </p>

<div class="alert alert–success" style="color:Black; margin–top: –16px; margin–right: 3px;">Пульс: {{ this.pulse }}</div>

</article>

{{/each}}

</section>

<canvas id="myChart"></canvas>

<canvas id="myChart2"></canvas>

</div>

<div class="modal–footer">

<button type="button" class="btn btn–secondary" data–dismiss="modal">Закрыть</button>

</div>

</div>

</div>

</div>

–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––

<div class="modal fade" id="exampleModal" tabindex="–1" role="dialog" aria–labelledby="exampleModalLabel" aria–hidden="true">

<div class="modal–dialog" role="document">

<div class="modal–content">

<div class="modal–header">

<h5 class="modal–title" id="exampleModalLabel">Данные пациента</h5>

<button type="button" class="close" data–dismiss="modal" aria–label="Close">

<span aria–hidden="true">&times;</span>

</button>

</div>

<div class="modal–body">

<form method="post" action="/card">

<div class="form–group">

<label for="recipient–name" class="col–form–label">ФИО</label>

<input type="text" class="form–control" id="recipient–name" name="FIO" required>

</div>

<div class="form–group">

<label for="recipient–name" class="col–form–label" >Дата рождения</label>

<input type="date" class="form–control" id="recipient–date" name="Datebirthday">

</div>

<div class="form–group">

<label for="recipient–name" class="col–form–label" >Дата госпитализации</label>

<input type="date" class="form–control" id="recipient–date–gos" name="Dategospital">

</div>

<div class="form–group">

<label for="exampleFormControlSelect1">Выберите номер датчика</label>

<select class="form–control" id="exampleFormControlSelect1" name="Datchik" required>

<Option>1</option>

<option>2</option>

</select>

</div>

<div class="form–group">

<label for="message–text" class="col–form–label" >История болезни</label>

<textarea class="form–control" id="message–text" name="History"></textarea>

</div>

<div class="form–group">

<label for="message–text" class="col–form–label" >Диагноз</label>

<textarea class="form–control" id="message–text" name="Diagnoz"></textarea>

</div>

</form>

</div>

<div class="modal–footer">

<button type="button" class="btn btn–secondary" data–dismiss="modal">Закрыть</button>

<button type="submit" class="btn btn–primary">Сохранить</button>

</div>

</div>

</div>

</div>

<script src="https://code.jquery.com/jquery–3.3.1.slim.min.js" integrity="sha384–q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384–UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384–JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

var temperature1 = 26.3,

temperature2 = 26.2,

temperature3 = 26.2,

temperature4= 26.2,

temperature5= 26.5,

temperature6= 26.5,

temperature7= 26.7,

temperature8= 26.7,

temperature9= 26.8,

temperature10= 26.8,

temperature11= 26.8,

temperature12= 26.8,

temperature13= 26.9,

temperature14 = 26.9,

temperature15= 26.9,

temperature16= 26.9,

temperature17= 26.9,

temperature18= 27.2,

temperature19= 27.2,

temperature20= 27.2,

temperature21= 27.3,

temperature22= 27.3,

temperature23= 27.3,

temperature24= 27.5;

var ctx = document.getElementById('myChart').getContext('2d');

var chart = new Chart(ctx, {

// The type of chart we want to create

type: 'line',

// The data for our dataset

data: {

labels: ['00:30', '01:00', '01:30', '02:00', '02:30', '03:00','03:30','04:00','04:30','05:00','05:30','06:00','06:30','07:00','07:30','08:00','08:30','09:00','09:30','10:00','10:30','11:00','11:30',

'12:00'],

datasets: [{

label: 'My First dataset',

backgroundColor: 'rgb(255, 99, 132)',

borderColor: 'rgb(255, 99, 132)',

data: [temperature1, temperature2, temperature3, temperature4, temperature5, temperature6, temperature7, temperature8, temperature9, temperature10, temperature11, temperature12, temperature13, temperature14, temperature15, temperature16, temperature17, temperature18, temperature19, temperature20, temperature21, temperature22, temperature23, temperature24]

}]

},

// Configuration options go here

options: {}

});

</script>

<script>

var temperature25= 27.5,

temperature26= 27.5,

temperature27= 27.6,

temperature28= 27.6,

temperature29= 27.6,

temperature30= 27.7,

temperature31= 27.7,

temperature32= 27.7,

temperature33= 27.8,

temperature34= 27.8,

temperature35= 27.9,

temperature36= 27.9,

temperature37== 27.9,

temperature38= 28.2,

temperature39= 28.2,

temperature40= 28.6,

temperature41= 28.6,

temperature42= 28.9,

temperature43= 28.9,

temperature44= 29,

temperature45= 29.5,

temperature46= 29.5,

temperature47= 30,

temperature48= 30.2;

var ctx = document.getElementById('myChart2').getContext('2d');

var chart = new Chart(ctx, {

// The type of chart we want to create

type: 'line',

// The data for our dataset

data: {

labels: ['12:30','13:00','13:30','14:00','14:30','15:00','15:30','16:00','16:30','17:00','17:30','18:00','18:30','19:00','19:30','20:00','20:30','21:00','21:30','22:00','22:30','23:00','23:30', '00:00'],

datasets: [{

label: 'My First dataset',

backgroundColor: 'rgb(255, 99, 132)',

borderColor: 'rgb(255, 99, 132)',

data: [temperature25, temperature26, temperature27, temperature28, temperature29, temperature30, temperature31, temperature32,temperature33, temperature34, temperature35, temperature36,temperature37, temperature38, temperature39, temperature40, temperature41, temperature42, temperature43, temperature44, temperature45, temperature46, temperature47, temperature48]

}]

},

// Configuration options go here

options: {}

});

</script>

</body>

</html>

<!DOCTYPE html>

<html >

<head>

<meta charset="UTF–8">

<title>Diplom</title>

<link rel="stylesheet" href="/css/style.css">

</head>

<body>

<div class="login–page">

<div class="form">

<div style="margin–top: 18px;">

<form action="/register" class="register–form" method="post" >

<input type="text" name="username" placeholder="Имя пользователя"/>

<input type="password" name="password" placeholder="Пароль"/>

<input type="text" name="FIO" placeholder="ФИО"/>

<input type="text" name="Rang" placeholder="Должность"/>

<button type="submit">Создать</button>

<p class="message">Уже зарегестрированы? <a href="#">Войти</a></p>

<% if (errors != null && showRegisterForm != null && showRegisterForm === true) { %>

<% errors.forEach(function(err){ %>

<p class="message error"><%= err %></p>

<% }); %>

<% } %>

</form>

<form action="/login" class="login–form" method="post">

<input type="text" name="username" placeholder="Имя пользователя"/>

<input type="password" name="password" placeholder="Пароль"/>

<button type="submit">Войти</button>

<p class="message">Не зарегестрированы? <a href="#">Создать аккаунт</a></p>

<% if (errors != null && showRegisterForm == null) { %>

<% errors.forEach(function(err){ %>

<p class="message error"><%= err %></p>

<% }); %>

<% } if (success != null && showRegisterForm == null) { %>

<p class="message success"><%= success %></p>

<% } %>

</form>

</div>

</div>

</div>

<script src='http://cdnjs.cloudflare.com/ajax/libs/jquery/3.1.0/jquery.min.js'></script>

<% if(showRegisterForm != null && showRegisterForm === true) { %>

<script>

$('.register–form').show(); $('.login–form, .form–line').hide();

</script>

<% } %>

<script>

$('.message a').click(function(){

$('form, .social–buttons, .form–line').animate({height: "toggle", opacity: "toggle"}, "slow");

});

</script>

</body>

</html>

––––––––––––––––––––––––––––––––––––––––––––––––

<!DOCTYPE html>

<html >

<head>

<meta charset="UTF–8">

<title>Diplom</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384–ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">

<link rel="stylesheet" href="/css/style.css">

</head>

<body>

<div id="wrapper">

<div id="header">

<a href="/card" class="btn btn–primary" style="border: 10px; border–color: rgb(128,137,158); ">Пациенты</a>

<a href="/worker" class="btn btn–primary" style="border: 10px; border–color: rgb(128,137,158); ">Работники</a>

<a href="/logout" class="btn btn–primary" style="border: 10px; border–color: rgb(128,137,158); ">Выйти</a>

</div>

<div id="left–sidebar"></div>

<div id="content">

<div id="inner–content">

</div>

</div>

<div id="footer"></div>

</div>

</body>

</html>

– –––––––––––––––– – – – ––––––––––––––––––– – – –

<!DOCTYPE html>

<html >

<head>

<meta charset="UTF–8">

<title>Diplom</title>

<link rel="stylesheet" href="/css/style.css">

<link rel='stylesheet prefetch' href='https://maxcdn.bootstrapcdn.com/font–awesome/4.4.0/css/font–awesome.min.css'>

</head>

<body>

<div class="container clearfix" style="width: 500px;">

<div class="controls" style="border: none; width: 100%;">

<a href="/logout" class="logout–btn" style="width: auto; padding: 13px; font–size: 12px;">Выйти</a>

</div>

<div class="room" style="width: 100%;">

</div>

<script src='http://cdnjs.cloudflare.com/ajax/libs/jquery/3.1.0/jquery.min.js'></script>

<script src="/socket.io/socket.io.js"></script>

<script src="/js/main.js"></script>

<script>$(function(){app.rooms()});</script>

</body>

</html>

––––––––––––––––––––––––––––––––––––––––––––––

<!DOCTYPE html>

<html >

<head>

<meta charset="UTF–8">

<title>Diplom</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384–ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">

<link rel="stylesheet" href="/css/style.css">

</head>

<body>

<div id="wrapper">

<div id="header">

<a href="/main" class="btn btn–primary" style="border: 10px; border–color: rgb(128,137,158); ">Главная</a>

<a href="/card" class="btn btn–primary" style="border: 10px; border–color: rgb(128,137,158); ">Пациенты</a>

<a href="/logout" class="btn btn–primary" style="border: 10px; border–color: rgb(128,137,158); ">Выйти</a>

</div>

<div id="left–sidebar"></div>

<div id="content">

<div id="inner–content">

<button type="button" class="btn btn–primary" data–toggle="modal" data–target="#exampleModal" style="border: 10px; border–color: rgb(128,137,158);">{{# us2}} {{FIO}} {{/us2}}</button>

<button type="button" class="btn btn–primary" data–toggle="modal" data–target="#exampleModal1" style="border: 10px; border–color: rgb(128,137,158);">{{# us}} {{FIO}} {{/us}}</button>

</div>

</div>

<div id="footer">

<div class="modal fade" id="exampleModal1" tabindex="–1" role="dialog" aria–labelledby="exampleModalLabel" aria–hidden="true">

<div class="modal–dialog" role="document">

<div class="modal–content">

<div class="modal–header">

<h5 class="modal–title" id="exampleModalLabel">{{# us}} {{FIO}} {{/us}}</h5>

<button type="button" class="close" data–dismiss="modal" aria–label="Close">

<span aria–hidden="true">&times;</span>

</button>

</div>

<div class="modal–body">

<section class="get">

{{# us }}

<article class="us" style="">

<div class="alert alert–success" style="color:Black; width: 100%;">Имя пользователя: {{ this.username }}</div>

<div class="alert alert–success" style="color:Black; margin–top: –16px; width: 100%;">ФИО: {{ this.FIO }}</div>

<div class="alert alert–success" style="color:Black; margin–top: –16px; width: 100%;">Должность: {{ this.Rang }}</div>

<br></br>

</article>

{{/us}}

</section>

</div>

</div>

</div>

</div>

<div class="modal fade" id="exampleModal" tabindex="–1" role="dialog" aria–labelledby="exampleModalLabel" aria–hidden="true">

<div class="modal–dialog" role="document">

<div class="modal–content">

<div class="modal–header">{{# us2}} {{FIO}} {{/us2}}</h5>

<button type="button" class="close" data–dismiss="modal" aria–label="Close">

<span aria–hidden="true">&times;</span>

</button>

</div>

<div class="modal–body">

<section class="get">

{{# us2 }}

<article class="us2" style="">

<div class="alert alert–success" style="color:Black; width: 100%;">Имя пользователя: {{ this.username }}</div>

<div class="alert alert–success" style="color:Black; margin–top: –16px; width: 100%;">ФИО: {{ this.FIO }}</div>

<div class="alert alert–success" style="color:Black; margin–top: –16px; width: 100%;">Должность: {{ this.Rang }}</div>

<br></br>

</article>

{{/us2}}

</section>

</div>

</div>

</div>

<script src="https://code.jquery.com/jquery–3.3.1.slim.min.js" integrity="sha384–q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384–UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384–JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>

<script type="text/javascript">$

$('#exampleModal').on('show.bs.modal', function (event) {

var button = $(event.relatedTarget) // Button that triggered the modal

var recipient = button.data('whatever') // Extract info from data–\* attributes

// If necessary, you could initiate an AJAX request here (and then do the updating in a callback).

// Update the modal's content. We'll use jQuery here, but you could use a data binding library or other methods instead.

var modal = $(this)

modal.find('.modal–title').text('New message to ' + recipient)

modal.find('.modal–body input').val(recipient)

})

</script>

</body>

</html>

{

"name": "Web",

"version": "1.0.0",

"description": "Диплом",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"start": "node server.js",

"dem" : "nodemon server.js"

},

"author": "Строкин Никита",

"dependencies": {

"bcrypt–nodejs": "0.0.3",

"body–parser": "^1.15.2",

"chart.js": "^2.8.0",

"connect–flash": "^0.1.1",

"connect–mongo": "^1.3.2",

"ejs": "^2.5.1",

"express": "^4.14.0",

"express–session": "^1.14.0",

"hbs": "^4.0.4",

"mongoose": "^5.0.17",

"nodemailer": "^6.2.1",

"nodemon": "^1.19.1",

"passport": "^0.3.2",

"passport–local": "^1.0.0",

"redis": "^2.6.2",

"socket.io": "^2.0.3",

"socket.io–redis": "^5.1.0",

"winston": "^2.2.0",

"xoauth2": "^1.2.0"

}

}

@import url(https://fonts.googleapis.com/css?family=Roboto:300);

\*, \*:before, \*:after {

box–sizing: border–box;

}

#header {

background–color: #dddddd;

}

#footer {

clear: both;

background–color: #dddddd;

}

#left–sidebar {

width: 200px;

height: 1000px;

float: left;

background–color: #dddddd;

position: relative;

z–index: 2;

}

#content {

background–color: #eeeeee;

float: left;

height: 1000px;

width: 100%;

box–sizing: border–box;

margin–left: –200px;

padding–left: 200px;

}

body {

background: #f8f8ff;

font–family: "Roboto", sans–serif;

–webkit–font–smoothing: antialiased;

–moz–osx–font–smoothing: grayscale;

}

.container {

margin: 2% auto;

width: 760px;

background: #444753;

}

.align–left {

text–align: left;

}

.align–right {

text–align: right;

}

.float–right {

float: right;

}

.clearfix:after {

visibility: hidden;

display: block;

font–size: 0;

content: " ";

clear: both;

height: 0;

}

.login–page {

width: 400px;

padding: 8% 0 0;

margin: auto;

}

.form {

position: relative;

z–index: 1;

background: #FFFFFF;

max–width: 400px;

margin: 0 auto 100px;

padding: 45px;

text–align: center;

box–shadow: 0 0 20px 0 rgba(0, 0, 0, 0.2), 0 5px 5px 0 rgba(0, 0, 0, 0.24);

}

.form input {

font–family: "Roboto", sans–serif;

outline: 0;

background: #f2f2f2;

width: 100%;

border: 0;

margin: 0 0 15px;

padding: 15px;

box–sizing: border–box;

font–size: 14px;

}

.form button {

font–family: "Roboto", sans–serif;

text–transform: uppercase;

outline: 0;

background: #86BB71;

width: 100%;

border: 0;

padding: 15px;

color: #FFFFFF;

font–size: 14px;

–webkit–transition: all 0.3 ease;

transition: all 0.3 ease;

cursor: pointer;

}

.form button:hover,.form button:active,.form button:focus {

background: #89c771;

}

.form .register–form {

display: none;

}

.loginBtn {

width: 80%;

}

.form {

background–color: #F2F5F8;

}

.form input {

background–color: white;

}

.form–line {

position: absolute;

left: 0px;

width: 360px;

margin–top: 2px;

border–bottom: 2px solid white;

}

button:hover, button:active, button:focus {

background: #89c771;

}

a.logout–btn{

background–color: rgb(226, 98, 92);

}

a.logout–btn:hover, a.logout–btn:active, a.logout–btn:focus {

background–color: rgb(245, 106, 99);

}

.controls {

width: 270px;

float: left;

margin: 0;

text–align: center;

border–bottom: 2px solid white;

}

.controls a {

width: 50%;

margin: 0;

}

.controls a {

text–decoration: none;

}

.container {

overflow: hidden;

}

.not–found–error {

padding: 5%;

background–color: white;

}

'use strict';

var app = ({

});