МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Гродненский государственный университет имени Янки Купалы»

Факультет математики и информатики

Кафедра современных технологий программирования

МЯЛИК ЕВГЕНИЙ АЛЕКСАНДРОВИЧ

ОТЧЕТ  
О выполнении программы ПРЕДДИПЛОМНОЙ ПРАКТИКи

студента 4 курса специальности   
1-40 01 01 «Программное обеспечение информационных технологий»

База практики: *ООО «Алеста Логистик Гроуп»*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Вид практики: производственная

Период прохождения практики: 18.03.2024 - 13.04.2024

Руководитель от кафедры:   
Урбан О.И. старший преподаватель кафедры СТП

\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись)*

**Утверждаю** 12.04.2024 г.

должность: Директор

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Мартинкевич С. А. *(подпись)* М.П.

**Гродно, 2024**

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc164333105)

[ГЛАВА 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc164333106)

[1.1 Описание предметной области 4](#_Toc164333107)

[1.2 Обзор существующих решений 5](#_Toc164333108)

[1.2.1 Сайт Гомельской областной клинической психиатрической больницы 5](#_Toc164333109)

[1.3 Вывод по главе 1 7](#_Toc164333110)

[ГЛАВА 2 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОГО САЙТА 8](#_Toc164333111)

[**2.1 Использование необходимых технологий разработки** 8](#_Toc164333112)

[**2.2 Проектирование внешнего вида сайта** 9](#_Toc164333113)

[**2.3 Структура клиентской части приложения** 10](#_Toc164333114)

[**2.4 Вывод по главе 2** 11](#_Toc164333115)

[ГЛАВА 3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СЕРВЕРНОЙ ЧАСТИ ИНФОРМАЦИОННОГО САЙТА 12](#_Toc164333116)

[**3.1 База данных** 12](#_Toc164333117)

[**3.2 Общая архитектура серверного приложения** 13](#_Toc164333118)

[**3.2.1 Controllers** 15](#_Toc164333119)

[**3.2.2 Routes** 18](#_Toc164333120)

[**3.2.3 Models** 19](#_Toc164333121)

[**3.2.4 Middleware** 20](#_Toc164333122)

[**3.3 Выводы по главе 3** 21](#_Toc164333123)

[ЗАКЛЮЧЕНИЕ 22](#_Toc164333124)

# ВВЕДЕНИЕ

Психиатрическая больница – место где врачи проводят обследования, лечение, реабилитацию и профилактику при нарушениях в психической сфере: от легких (неврозы, реактивные состояния, синдром хронической усталости, расстройства личности) до тяжелых (психозы, слабоумие, попытки самоубийства, алкогольная и наркотическая зависимость).

В таких больницах проводится не только лечебная помощь, но и осуществляются консультации психиатров и психотерапевтов при подозрении на наличие психического заболевания, для исключения психиатрического диагноза, определения предрасположенности к нарушениям поведения.

К сожалению, не у всех существует возможность самостоятельно прийти в подобное учреждение чтобы записаться на приём или, например, узнать какую-то информацию. Именно поэтому в нынешнее время создаются веб-сайты, содержащие в себе все необходимые данные.

На сегодняшний день существует множество инструментов для реализации веб-сайтов. Наш выбор пал на JavaScript, а именно React для frontend части приложения, так как React упрощает процесс создания скриптов, что в свою очередь облегчает техническое обслуживание и повышает производительность, а также гарантирует более быстрый рендеринг. Также для вёрстки самого сайта был выбран популярный фреймворк Bootstrap, который позволяет упростить работу с вёрсткой. И Node.js для backend части, по той причине, что в Node.js есть возможность использовать npm — самый большой пакетный менеджер, а также он использует асинхронную модель выполнения, что позволяет эффективно обрабатывать множество запросов одновременно (до 10000 активных соединений) без блокировки. Для реализации СУБД был выбран PostgresSql так как данная СУБД банально имеет больший функционал, а также является более понятной и удобной по сравнению с остальными СУБД.

# ГЛАВА 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

# Описание предметной области

Целью прохождения преддипломной практики является подготовить front-end часть для информационного сайта УЗ «Психиатрия-наркология». Для начала мы должны были создать прототип сайта, чтобы иметь представление, какой вид приложение будет иметь в будущем. Сам сайт будет иметь перечисленные страницы:

* Главная страница
* Структура
* Новости коллектива
* Вакансии
* Услуги
* Обращения граждан
* Контакты

Также сайт будет иметь личный кабинет пользователя и различные роли, а именно:

* Незарегистрированный пользователь
* Пользователь
* Модератор
* Администратор

Администратор и модератор будут иметь возможность добавлять различные элементы на сайт (новости, публикации, фотографии в галерею и т.д.) при помощи админ-панели.

После создания прототипа мы начинаем создавать его макет. Макет отличается от прототипа тем, что является завершающий этапом представления внешнего вида сайта. После создания и согласования макета можно приступать уже к работе над самим приложением. По началу разработки устанавливаем все необходимые технологии в нашу среду разработки. После этого мы создаём структуру приложения, далее идёт настройка навигации и роутинга по сайту и создание глобального стейта для хранения информации. После этих этапов можно приступать к вёрстке самого сайта, попутно прописав функционал авторизации/регистрации пользователя и создание модальных окон. Далее идёт подключение к back-end части приложения и взаимодействие с нею.

# Обзор существующих решений

# 1.2.1 Сайт Гомельской областной клинической психиатрической больницы

На первый взгляд сайт выглядит вполне нормально, но опустившись ниже на главной странице можно увидеть максимально простое оформление, не имеющее нормально разделение элементов, а также маленький и не самый приятный на вид текст.

Но также можно подметить, что все кнопки работают так, как надо, а некоторые даже имеют нестандартный вид и изменяют свою внешность при наведении на них курсором.

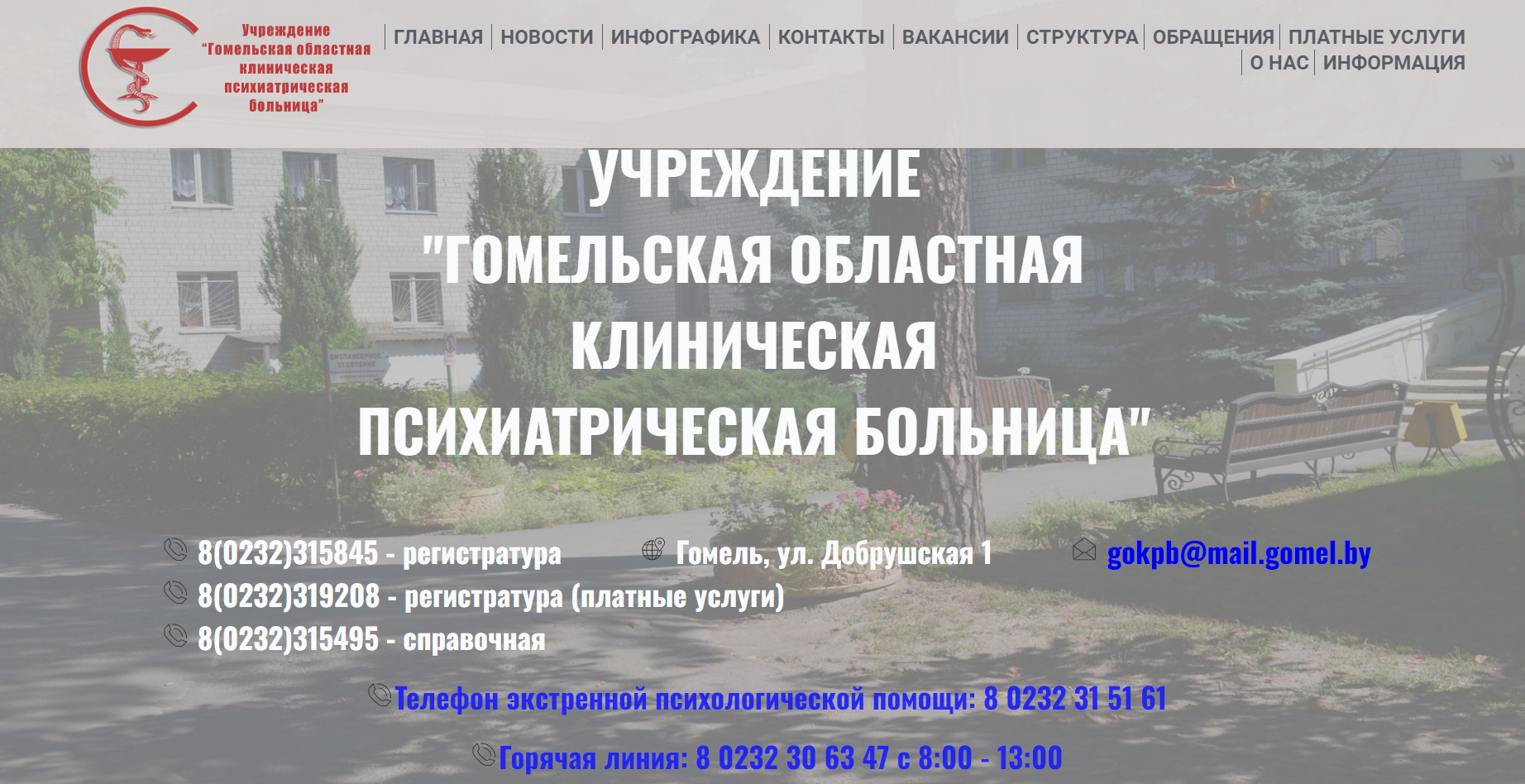


Рисунок 1.2 Главная страница сайта учреждение “Гомельская областная клиническая психиатрическая больница”

**1.2.2 Психиатрическая клиника Василия Шурова**

Совершенно другой случай по сравнению с ранее описанным сайтом. Очень приятное оформление, каждая кнопка кликабельна и меняет внешний вид при наведении на неё курсором. Информация также избыточна, но подано всё это в куда более приятном виде. Ещё можно отметить возможность изменять главную тему сайта на тёмную или светлую соответственно, маленькая вещь, но тоже приятно. Также у сайта есть редкая, но очень нужная для некоторых людей опция, а именно возможность менять шрифт, цвет и другие различные элементы сайта для людей со слабым зрением.

Справедливости ради стоит еще уточнить, что это сайт частной клиники, а не государственной, соответственно и бюджеты у подобных сайтов будут гораздо выше.

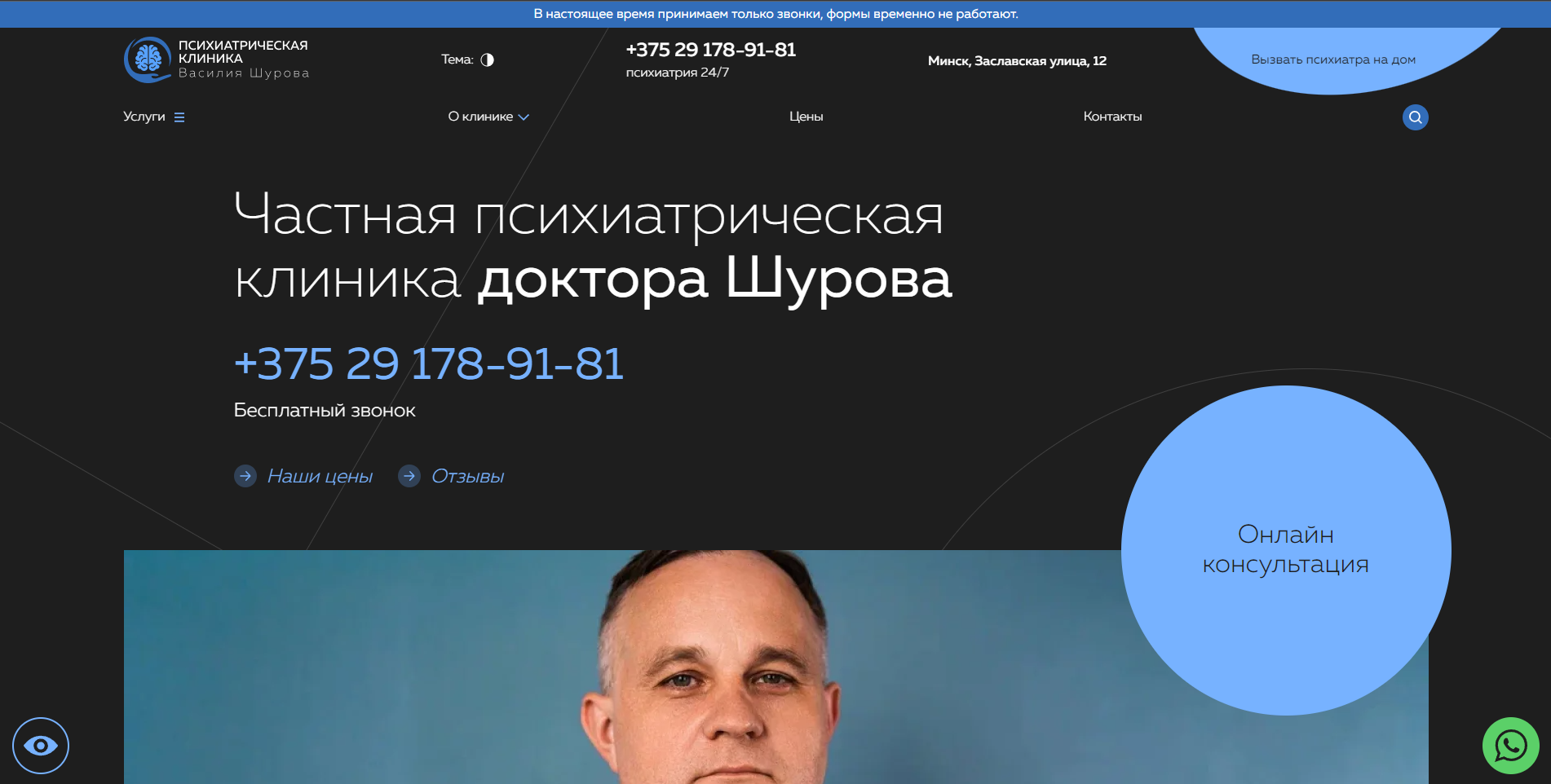
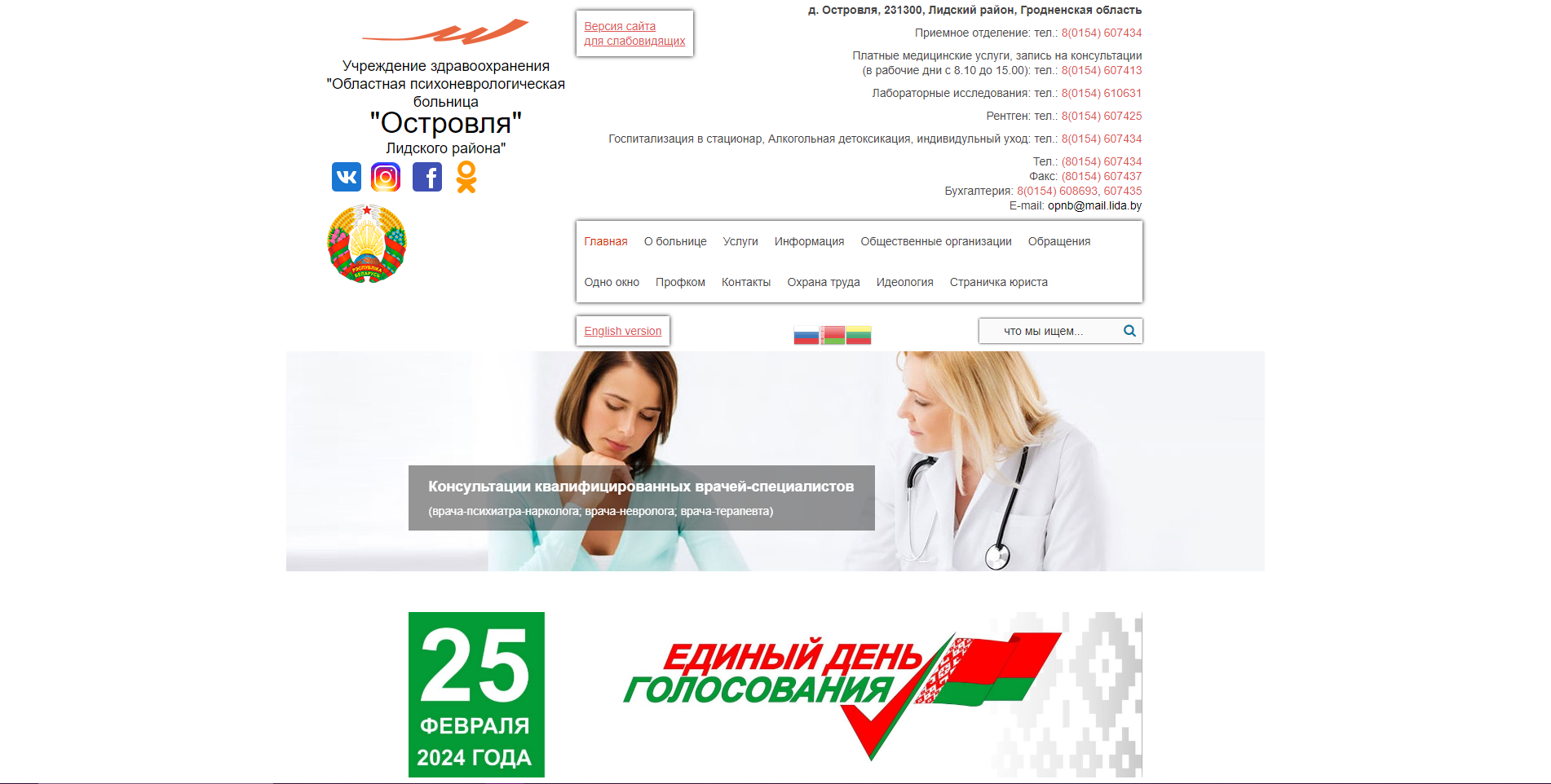


Рисунок 1.2.1 Главная страница сайта психиатрической клиники Василия Шурова

**1.2.3 УЗ “Областная психоневрологическая больница “Островля” Лидского района”**

Сразу бросается в глаза простое, но при этом хорошее оформление, без лишних элементов и текста, а также меняющиеся изображения на главной странице. Все кнопки кликабельны и выполняют нужный функционал, еще некоторые из них имеют плавное появление окошка с дополнительной информацией. У сайта имеется смена локализации на разные языки, а именно на русский, английский, белорусский и литовский. Также имеется переключение версии сайта на версию для слабовидящих, что определенно является плюсом. Вся опубликованная информация избыточна и понятна для чтения.

Рисунок 1.2.2Главная страницасайтаУЗ “Областная психоневрологическая больница “Островля” Лидского района”

# Вывод по главе 1

В первой главе была описана предметная область, были определены параметры для разных объектов и ограничения, которые будут на них накладываться. Так же были рассмотрены существующие решения, проанализированы их плюсы и минусы.

# ГЛАВА 2 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОГО САЙТА

**2.1 Использование необходимых технологий разработки**

Серверная часть приложения будет разработана с помощью Node.js. Для реализации будут использованы следующие технологии:

* Express
* Cors
* Sequelize
* Path

**Express** - это минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений.

**Cross-Origin Resource Sharing (**[**CORS**](https://developer.mozilla.org/ru/docs/Glossary/CORS)**)** — механизм, использующий дополнительные [HTTP](https://developer.mozilla.org/ru/docs/Glossary/HTTP)-заголовки, чтобы дать возможность [агенту пользователя](https://developer.mozilla.org/ru/docs/Glossary/User_agent) получать разрешения на доступ к выбранным ресурсам с сервера на источнике (домене), отличном от того, что сайт использует в данный момент. Говорят, что агент пользователя делает запрос с другого источника (cross-origin HTTP request), если источник текущего документа отличается от запрашиваемого ресурса доменом, протоколом или портом.

В целях безопасности браузеры ограничивают cross-origin запросы, инициируемые скриптами. Например, [XMLHttpRequest](https://developer.mozilla.org/ru/docs/Web/API/XMLHttpRequest) и [Fetch API](https://developer.mozilla.org/ru/docs/Web/API/Fetch_API) следуют *политике одного источника* ([same-origin policy](https://developer.mozilla.org/ru/docs/Web/Security/Same-origin_policy)). Это значит, что web-приложения, использующие такие API, могут запрашивать HTTP-ресурсы только с того домена, с которого были загружены, пока не будут использованы CORS-заголовки.

**Sequelize** - это ORM-библиотека для приложений на Node.js, которая осуществляет сопоставление таблиц в бд и отношений между ними с классами. При использовании Sequelize мы можем не писать SQL-запросы, а работать с данными как с обычными объектами. Причем Sequelize может работать с рядом СУБД - MySQL, Postgres, MariaDB, SQLite, MS SQL Server.

Модуль **node:path** предоставляет утилиты для работы с путями к файлам и каталогам.

**2.2 Проектирование внешнего вида сайта**

Figma (Фигма) — это графический онлайн-редактор для совместной работы. В нём можно создать прототип сайта, интерфейс приложения и обсудить правки с коллегами в реальном времени.

Используя Фигму мы создадим прототип сайта, на котором будет находиться примерное расположение элементов страницы, их размер и т.д.

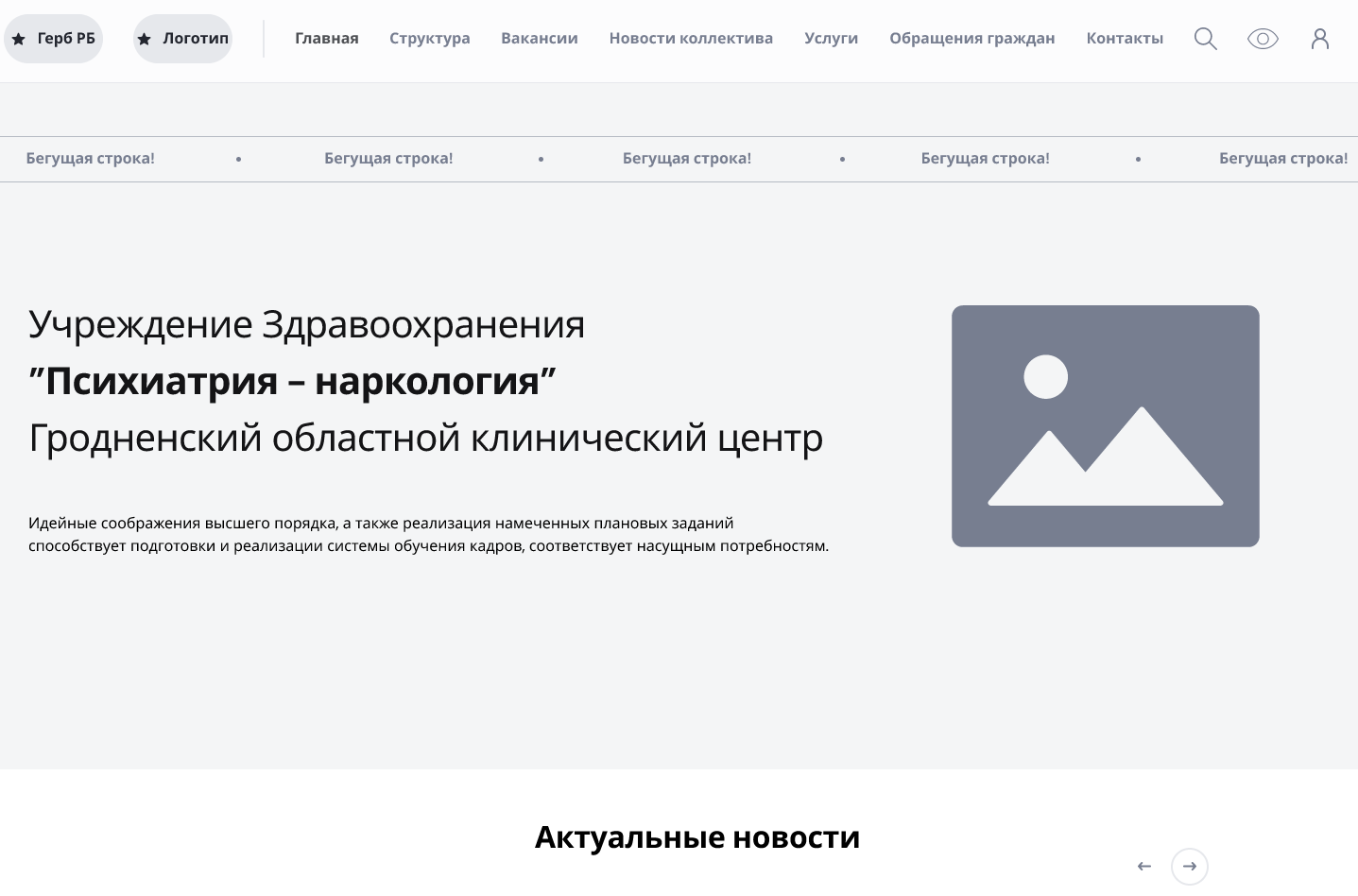


Рисунок 2.2.1 Прототип главной страницы

Так как прототип выступает в роли примера для внешнего вида сайта, то уже сам макет можно отличаться от него, но при этом стараясь сохранить расположение всех элементов.

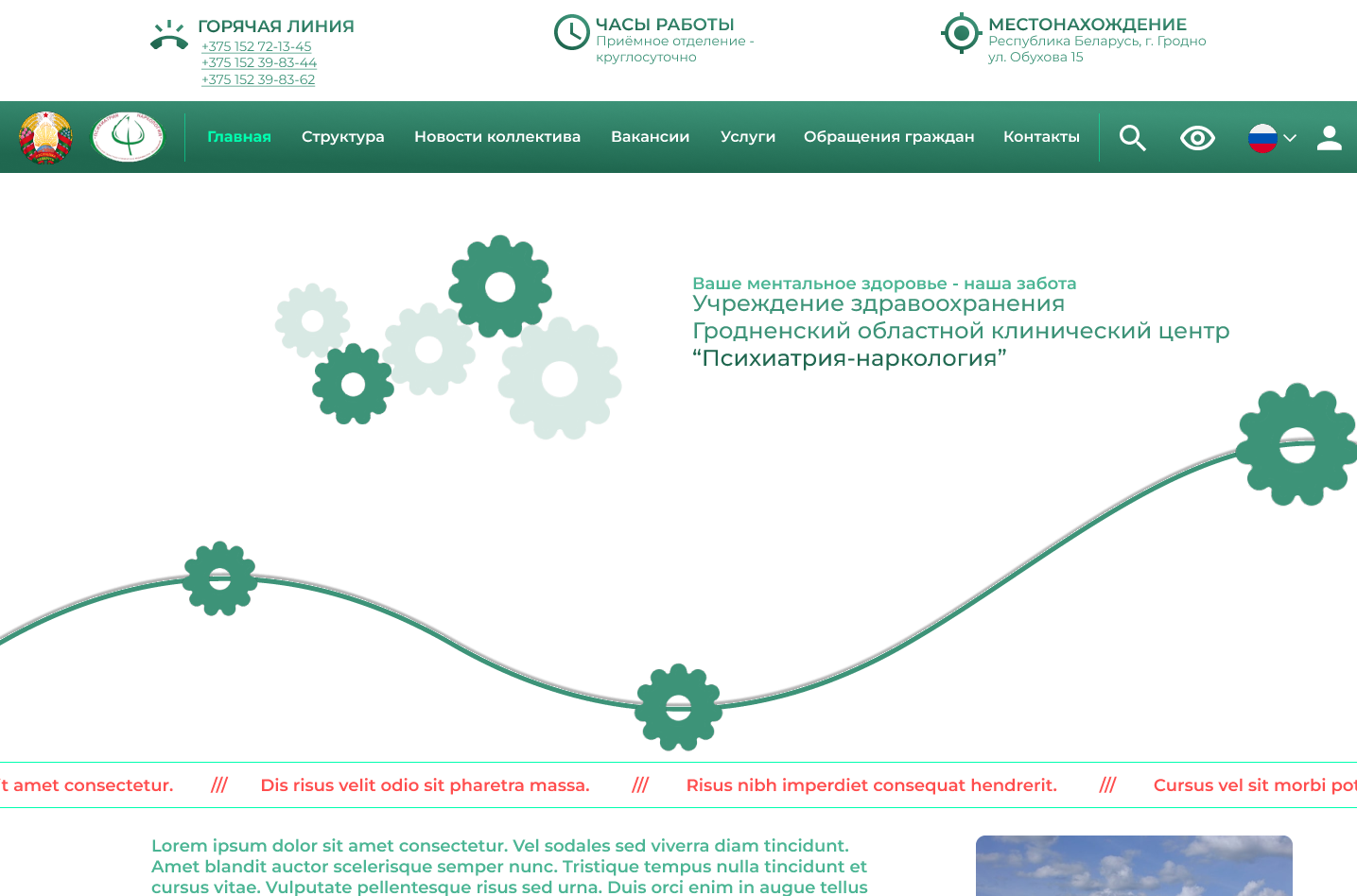


Рисунок 2.2.2 Макет главной страницы

Макет уже имеет в себе все нужные стили и шрифты для последующего начала разработки самого приложения, так как разработчик будет создавать внешний вид сайта полностью основываясь на готовом макете.

**2.3 Структура клиентской части приложения**

Для начала создаём все необходимые папки, а именно:

* server – в ней будут хранится все необходимые модули и объекты
* models – папка, в которой описаны все модели
* controllers – папка, содержащая контроллеры, соответствующие моделям
* routes – папка, содержащая маршруты, соответствующие моделям
* middleware – папка с миддлвейр

В самой папке server имеются такие файлы:

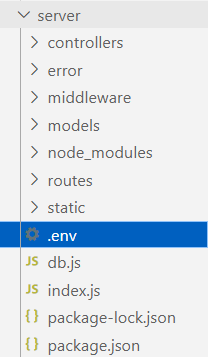
* index.js – описаны все используемые технологии
* db.js – подключение к базе данных
* .env – файл с фиксированной информацией, такой как порты и данные подключения к бд
* 

Рисунок 2.3 Структура приложения

**2.4 Вывод по главе 2**

Во второй главе были проанализированы и определены основные технологии разработки, разработан внешний вид сайта и создана общая структура приложения.

# ГЛАВА 3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СЕРВЕРНОЙ ЧАСТИ ИНФОРМАЦИОННОГО САЙТА

## **3.1 База данных**

База данных была создана с помощью средств СУБД pgAdmin 4 (PostgreSQL). Также присутствует реализация создания элементов в БД с помощью пользовательского интерфейса на примере таблицы users. (Рис 3.2).

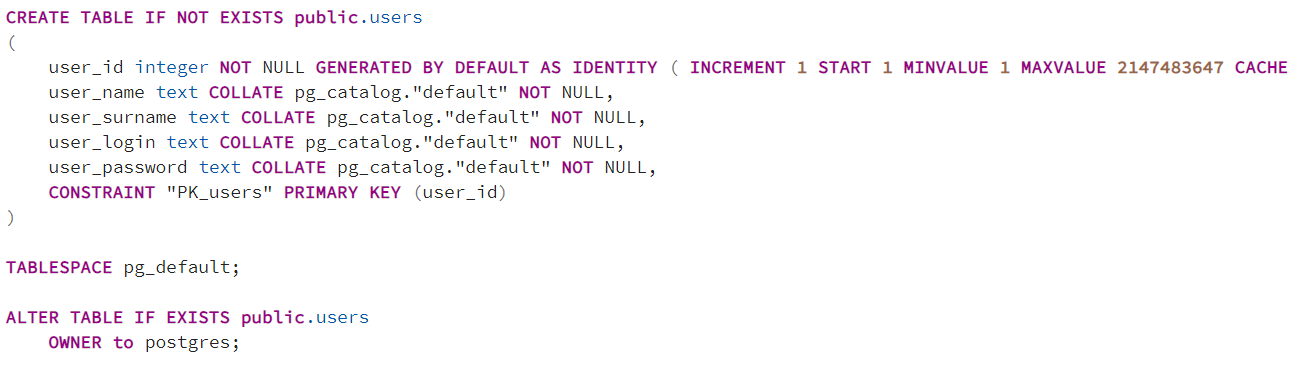


Рисунок 3.1 Таблица пользователей

Инициализация всех объектов происходит в основном index.js файле. Это модели, роуты, обработчики ошибок, а также все основные библиотеки, используемые для реализации приложения (Рис 3.2).



Рисунок 3.2 Инициализация

## **3.2 Общая архитектура серверного приложения**

Backend часть приложения реализована на языке программирования JavaScript с помощью фреймворка Express.

Express представляет собой популярный веб-фреймворк, написанный на JavaScript и работающий внутри среды исполнения node.js. Этот модуль освещает некоторые ключевые преимущества этого фреймворка, установку среды разработки и выполнение основных задач веб-разработки и развёртывания.

[Node](https://nodejs.org/) (или более формально *Node.js*) - кросплатформенная среда исполнения с открытым исходным кодом, которая позволяет разработчикам создавать всевозможные серверные инструменты и приложения используя язык [JavaScript](https://developer.mozilla.org/ru/docs/Glossary/JavaScript). Среда исполнения предназначена для использования вне контекста браузера (т.е. выполняется непосредственно на компьютере или на серверной ОС). Таким образом, среда исключает API-интерфейсы JavaScript для браузера и добавляет поддержку более традиционных OS API-интерфейсов, включая библиотеки HTTP и файловых систем.

С точки зрения веб-серверной разработки Node имеет ряд преимуществ:

* Отличная производительность! Node был разработан для оптимизации пропускной способности и масштабируемости в веб-приложениях и очень хорошо справляется со многими распространёнными проблемами веб-разработки (например, веб-приложения реального времени).
* Код написан на «обычном старом JavaScript», а это означает, что затрачивается меньше времени при написании кода для браузера и веб-сервера связанное с «переключением технологий» между языками.
* JavaScript является относительно новым языком программирования и имеет преимущества от улучшения дизайна языка по сравнению с другими традиционными языками для веб-серверов (например, Python, PHP, и т.д.). Многие другие новые и популярные языки компилируются/конвертируются в JavaScript, поэтому вы можете также использовать CoffeeScript, ClosureScript, Scala, LiveScript, etc.
* Менеджер пакетов Node (NPM) обеспечивает доступ к сотням тысяч многоразовых пакетов. Он также имеет лучшее в своём классе разрешение зависимостей и может также использоваться для автоматизации большинства инструментов построения.
* Он портативен, имеет версии для Microsoft Windows, OS X, Linux, Solaris, FreeBSD, OpenBSD, WebOS, и NonStop OS. Кроме того, он имеет хорошую поддержку среди многих хостинг-провайдеров, которые часто предоставляют конкретную инфраструктуру и документацию для размещения сайтов, работающих на Node.
* Он имеет очень активную стороннюю экосистему и сообщество разработчиков, которые всегда готовы помочь.

Будут реализованы следующие слои приложения:

### **3.2.1 Controllers**

Контроллер в осуществлении CRUD операций (Create, Read, Update, Delete) используется для управления всем процессом взаимодействия между пользователем и базой данных. Он отвечает за обработку запросов, полученных от пользователя и передачу этих запросов в модель (базу данных) для выполнения соответствующих операций. Контроллер также отвечает за формирование ответа пользователю на основе результатов выполнения операций в модели. Он также обеспечивает проверку данных, введенных пользователем, и обеспечивает безопасность приложения. В целом контроллер является ключевым компонентом в архитектуре приложения, которая позволяет управлять всем процессом взаимодействия между пользователем и базой данных. (Рис 3.3).

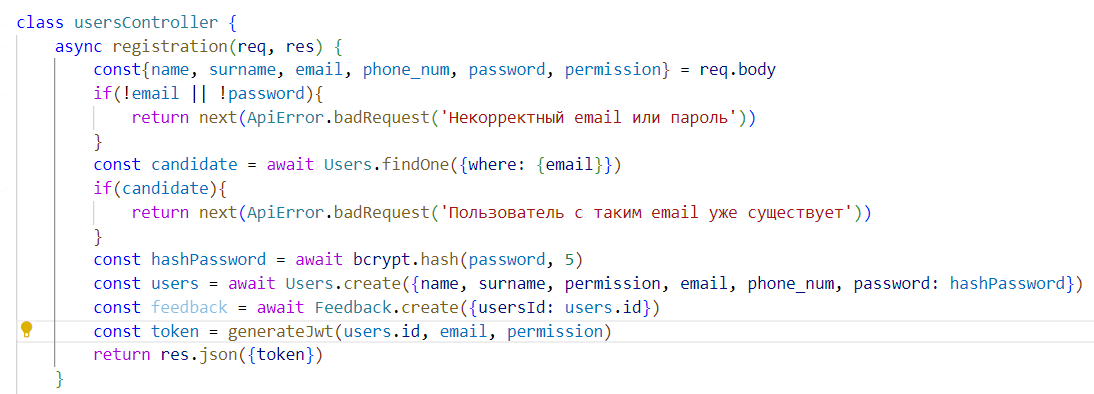


Рисунок 3.3 Контроллер для создания и заполнения новой таблицы пользователей с обработкой jwt-токена

При создания новой таблицы в Express с помощью контроллера выполняются следующие шаги: создаётся модель данных с помощью классов, которые будут представлять таблицу и ее свойства; создаётся контекст данных, который будет управлять взаимодействием с базой данных; создаётся контроллер, который будет обрабатывать запросы от пользователя и передавать их в контекст данных для выполнения операций над таблицей; используются методы контроллера для выполнения операций CRUD (Create, Read, Update, Delete) над таблицей; отправляются ответы пользователю на основе результатов выполнения операций в контексте данных; обеспечивается безопасность приложения путем проверки данных, введенных пользователем.

**JSON Web Token**, или просто **JWT**, представляет собой строку, полученную на основе формата JSON, и используется в качестве более безопасной и простой альтернативы сессиям и файлам cookie для авторизации.

JWT позволяет уйти от хранения данных авторизованного пользователя на сервере и возлагает на сервер только задачу по верификации подписи.

JWT формируют три части:

* заголовок (header);
* данные (payload);
* подпись (signature).

Заголовок представляет собой объект JSON и описывает сам токен с помощью следующих свойств:

* alg - алгоритм шифрования, используемый для подписи JWT, если токен не подписывается, то значением должно быть none (обязательный параметр);
* typ - тип токена, необходимо указывать со значением "JWT", если могут использоваться токены другого типа (необязательный параметр);
* ctp - тип данных, необходимо указывать со значением "JWT", если в payload присутствуют пользовательские ключи.

В данных, которые также передаются объектом JSON, указывается необходимая информация о пользователе. Также возможно задание значений предопределенных ключей (все они не обязательны) для описания конфигурации токена:

* iss - приложение, создавшее токен;
* sub - назначение JWT;
* aud - массив получателей токена;
* exp - дата и время, указанное в миллисекундах, прошедших с 01.01.1970, до наступления которого JWT будет валиден;
* nbf - дата и время, указанное в миллисекундах, прошедших с 01.01.1970, до наступления которого JWT будет не валиден;
* iat - дата и время создания JWT, указанное в миллисекундах, прошедших с 01.01.1970;
* jti - уникальный идентификатор токена.

Заголовок и данные используются для вычисления значения подписи по указанному в заголовке в свойстве alg алгоритму шифрования.

Метод генерации jwt-токена представлен на рисунке 3.4.

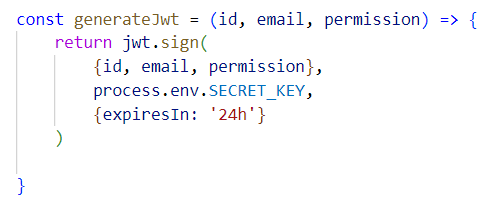


Рисунок 3.4 Метод генерации jwt-токена

**3.2.2 Routes**

С помощью **маршрутизации** описываются ответы Node.js приложения на запросы, поступающие с клиентской стороны или другого сервера (в случае, если реализуется API) на конкретный URL. Передаваемые данные и информация об отправителе запроса содержится в [объекта запроса](https://nodejsdev.ru/guides/webdraftt/request-object/), а с помощью [объекта ответа](https://nodejsdev.ru/guides/webdraftt/response-object/) формируется ответ на запрос. С помощью express.Router() можно описать Node.js маршрутизацию как систему взаимосвязанных Middleware (промежуточных обработчиков).



Рисунок 3.5 Маршруты

**3.2.3 Models**

Ключевым компонентом в работе Sequelize с бд являются модели. Модели описывают стуктуру хранящихся в бд данных. Кроме того, через модели в основном идет взаимодействие с бд.

Есть два способа определения моделей. Первый способ представляет использование метода define(). (Рис 3.6).

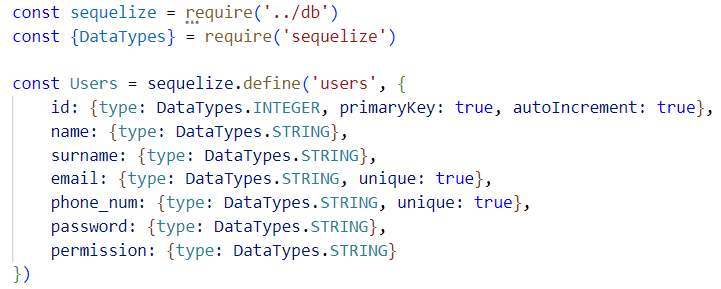


Рисунок 3.6 Модель пользователей

Дополнительно для поля можно задать еще ряд атрибутов. Атрибут **allowNull** указывает, допускает ли поле отсутствие значение. По умолчанию имеет значение true - то есть у поля может отсутствовать значение. Значение false указывает, что поле обязательно должно иметь какое-либо значение.

Для поля id в данном случае также задается еще пара атрибутов. Атрибут autoIncrement: true указывает, что значение столбца в таблице в БД будет автоинкрементироваться, а атрибут primaryKey: true говорит о том, что соответствующий столбец в таблице будет выполнять роль первичного ключа.

**3.2.4 Middleware**

Node.js middleware, или по-другому **функция промежуточной обработки**, используется для выполнения каких-либо действий на основе данных объекта запроса и ответа и передает обработку следующей функции. (Рис 3.7)



Рисунок 3.7 Middleware для проверки роли пользователя

## **3.3 Выводы по главе 3**

В главе о физической модели данных были рассмотрены основные аспекты работы с базой данных и Express.js. Было показано, как создавать таблицы и связи между ними. Также было рассмотрено создание обработчика jwt для реализации авторизации. В данный момент реализованы все основные моменты для серверной части приложения.

# ЗАКЛЮЧЕНИЕ

В ходе прохождения преддипломной практики был разработан основной каркас для информационного сайта при помощи языка программирования JavaScript и фреймворка Express. При разработке приложения были отработаны навыки по программированию на языке JavaScript и изучены возможности Node.js.

Был проведён анализ существующих решений, спроектирован прототип и макет веб-сайта, реализованы основные компоненты приложения, разобран каждый этап в ходе создания структуры веб-сайта и реализовано подключение к front-end части приложения.

Таким образом, все поставленные в работе цели были достигнуты.

**CПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. https://react-bootstrap.netlify.app – Дата доступа: 16.03.2024
2. https://habr.com/ru – Дата доступа: 05.04.2024
3. https://www.postgresql.org – Дата доступа: 15.03.2024
4. https://learn.javascript.ru – Дата доступа: 20.03.2024
5. https://ru.wikipedia.org/wiki/База\_данных – Дата доступа: 15.03.2024
6. <https://nodejs.org/en/learn/getting-started/introduction-to-nodejs> – Дата доступа: 25.03.2024