**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**Федеральное государственное бюджетное образовательное**

**учреждение высшего образования**

**«Пензенский государственный университет»**

(**ФГБОУ ВО «Пензенский государственный университет») \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Кафедра «Математическое обеспечение и применение ЭВМ»**

**Зав. Кафедрой *Макарычев П.П.***

**"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.**

**Приложение записи на прием к врачу для информационной системы медицинских организаций города**

Пояснительная записка к курсовому проекту по дисциплине

**«Базы данных»**

ПГУ 09.03.04 – 08КР151.04 ПЗ

Автор работы Бурлин В.В.

Группа 15ВП1

Направление 09.03.04

Руководитель работы Шибанов С.В.

Работа защищена "\_\_\_"\_\_\_\_\_\_\_\_2019 г.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**Федеральное государственное бюджетное образовательное**

**учреждение высшего образования**

**«Пензенский государственный университет»**

(**ФГБОУ ВО «Пензенский государственный университет») \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Кафедра «Математическое обеспечение и применение ЭВМ»**

**«УТВЕРЖДАЮ»**

**Зав. Кафедрой *Макарычев П.П.***

**7 сентября 2018 г.**

**ЗАДАНИЕ**

**на курсовое проектирование по дисциплине**

**«Базы данных»**

**Студенту Бурлину В.В. Группа 15ВП1**

**Тема проекта** Приложение записи на прием к врачу для информационной системы медицинских организаций города

**Исходные данные (технические требования) на проектирование**

1. Выделить категории пользователей разрабатываемой системы, провести анализ их потребностей, определить запросы для каждой категории пользователей.
2. Определить ограничения целостности.
3. Разрабатываемая подсистема должна быть построена в архитектуре «клиент-сервер», в качестве сервера базы данных использовать СУБД PostgreSQL 9.6 и выше.
4. Для реализации запросов к базе данных использовать объектно-реляционное отображение (ORM).
5. Программное обеспечение должно быть полностью отлажено и протестировано, и должно функционировать под управлением Mojave.

**1. Объем работы по курсу**

1. Анализ требований к разработке приложения записи на прием к врачу для информационной системы медицинских организаций города;
2. Проектирование базы данных приложения записи на прием к врачу для информационной системы медицинских организаций города;
3. Разработка программного обеспечения приложения записи на прием к врачу для информационной системы медицинских организаций города;

**2. Графическая часть**

1. Диаграмма вариантов использования приложения записи на прием к врачу для информационной системы медицинских организаций города;
2. Схема концептуальной модели БД с указанием имен сущностей и названий атрибутов, характеризующих эти сущности;
3. Диаграмма объектной модели БД с указанием имен сущностей, типов связей между ними и их мощности;
4. Диаграмма полной атрибутивной физической модели БД с указанием имен типов сущностей, типов связей и их мощности, атрибутов типов сущностей с выделением первичных и внешних ключей.

**3. Экспериментальная часть**

Разработка, отладка и тестирование программного обеспечения приложения записи на прием к врачу для информационной системы медицинских организаций города.

**Срок выполнения проекта по разделам**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | Анализ требований к разработке приложения записи на прием к врачу для информационной системы медицинских организаций города | к | 21 сентября | 2018 г. |
| 2 | Анализ предметной области и проектирование концептуальной модели БД | к | 19 октября | 2018 г. |
| 3 | Определение архитектуры приложения записи на прием к врачу для информационной системы медицинских организаций города | к | 2 ноября | 2018 г. |
| 4 | Разработка, отладка и тестирование программного обеспечения приложения записи на прием к врачу для информационной системы медицинских организаций города | к | 29 ноября | 2018 г. |
| 5 | Оформление пояснительной записки | к | 14 декабря | 2018 г. |

**Дата выдачи задания** 7 сентября 2018 г.

**Дата защиты проекта**  2019 г.

**Руководитель**  */С.В. Шибанов /*

**Задание получил** 7 сентября 2018 г.

**Студент**   */В.В. Бурлин/*

Реферат

Пояснительная записка содержит 43 листа, 7 рисунков, 1 таблицу, 6 использованных источников, 2 приложения.

БАЗА ДАННЫХ, ВЕБ-СЕРВЕР, SQL, REST API, ОБЪЕКТНО-РЕЛЯЦИОННОЕ ОТОБРАЖЕНИЕ, ФУНКЦИОНАЛЬНОЕ ТЕСТИРОВАНИЕ

Объектом исследования являются способы проектирования баз данных для эффективного взаимодействия с ней веб-приложения.

Целью курсовой работы является разработка Приложение записи на прием к врачу для информационной системы медицинских организаций города.

Осуществлено функциональное тестирование разработанного программного обеспечения для разных категорий пользователей, что позволяет судить о корректности разработанных программных средств.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ПГУ 09.03.04 – 08КР151.04 ПЗ | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум. | Подп. | Дата |
| Разраб. | | Бурлин В.В. |  |  | «Приложение записи на прием к врачу для информационной системы медицинских организаций города»  Пояснительная записка. | Лит. | | | Лист | Листов |
| Пров. | | Шибанов С.В. |  |  |  |  |  | 4 |  |
|  | |  |  |  | Группа 15ВП1 | | | | |
| Н. контр. | |  |  |  |
| Утв. | |  |  |  |
|  | |  |  |  |

Оглавление

[Введение 6](#_Toc7238111)

[1 Постановка задачи и анализ предметной области 7](#_Toc7238112)

[1.1 Постановка задачи на разработку программного обеспечения 7](#_Toc7238113)

[1.2 Анализ требований 7](#_Toc7238114)

[1.3 Анализ предметной области 10](#_Toc7238115)

[1.4 Выбор технологий и средств разработки 11](#_Toc7238116)

[2 Проектирование программных средств и струкур данных 14](#_Toc7238117)

[2.1 Проектирование объектной модели базы данных 14](#_Toc7238118)

[2.2 Структура программного обеспечения 15](#_Toc7238119)

[2.3 Модель интерфейса 16](#_Toc7238120)

[3 Реализация программных средств 18](#_Toc7238121)

[3.1 Компоненты системы 18](#_Toc7238122)

[3.2 Физическая модель базы данных 20](#_Toc7238123)

[4 Тестирование программных средств 22](#_Toc7238124)

[Заключение 24](#_Toc7238125)

[Список использованных источников 25](#_Toc7238126)

[Приложение A. Модель программного интерфейса 26](#_Toc7238127)

[Приложение Б. Модель графического интерфейса 33](#_Toc7238128)

# Введение

В данной курсовой работе рассматривается информационная система медицинских организаций города. Предназначение данной информационной системы заключется в предоставлении доступа к информации о медицинских организациях города и специалистах, работающих в них. Одной из возможностей системы является запись к специалисту на определенную дату и время.

Целью курсовой работы является проектирование структуры базы данных рассматриваемой информационной системы, а также создание веб-приложения для работы с ней. Серверная составляющая веб-приложения должна обеспечивать взаимодействие с клиентской части с сервером базы данных. Назначение клиенской составляющей заключается в отображении информации, хранящейся в базе данных информационнй системы.

В процессе выполнения курсовой работы необходимо осуществить постановку задачи и анализ требований, а также спроектировать, реализовать и протестировать программные средства рассматриваемой подсистемы.

# 1 Постановка задачи и анализ предметной области

## 1.1 Постановка задачи на разработку программного обеспечения

В рамках данной курсовой работы рассматривается информационная система медицинских организаций города. Необходимо разработать веб-приложение, позволяющее пользователю просматривать информацию о медицинских организациях города, о специалистах, работающих в этих организациях, а также осуществлять запись к специалисту, предварительно выбрав дату и время приема.

Необходимо также предусмотреть возможность регистрации пользователей в информационной системе. Просмотр информации должен быть доступен любому посетителю сайта, а запись на прием только аутентифицированным пользователям.

## 1.2 Анализ требований

В разрабатываемой системе необходимо выделить следующие категории пользователей:

* доктор;
* пациент;
* гость.

В зависимости от категории пользователя ему предоставляются различные функциональные возможности (рисунок 1).

Любому пользователю должна быть доступна информация о медицинских организациях и специалистах, работающих в них. Пациентам должны быть предоставлены возможности записи и отмены записи на прием к доктору. Доктора также должны иметь возможности записи и отмены записи на прием к другому специалисту. Кроме того, докторам должна быть предоставлена возможность запрещать запись к себе на определенное время, а также отменять установленный запрет.

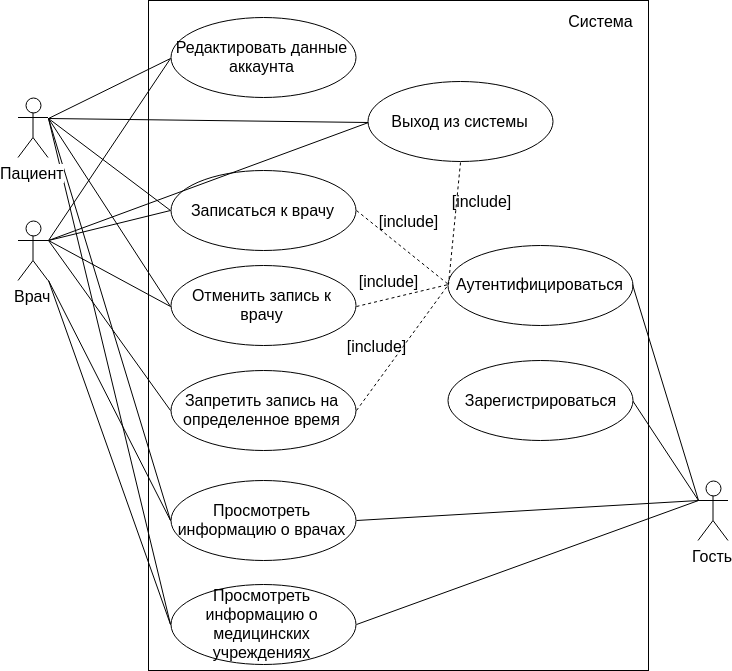


Рисунок – Функиональные возможности системы

Неаутентифицированные пользователи должны восприниматься системой как пользователи, относящиеся к категории «гость». Для перехода пользователя из категории «гость» в категорию «пациент» ему достаточно зарегистрироваться в системе, а затем пройти этап аутентификации. Учетные записи врачей должны выдаваться пользователям администраторами системы. Доктора и пациенты считаются аутентифицированными, вошедшими в систему пользователями, поэтому им должна быть предоставлена возможность выхода из системы. Пользователи, вошедшие в систему должны иметь возможность редактирования своих учетных записей.

Разрабатываемая система должна включать в свой состав интуитивно-понятный графический интерфейс пользователя. Должно быть созданы следующие страницы:

* главная страница;
* страница просмотра информации о медицинских организациях;
* страница просмотра информации о медицинской организации;
* страница выбора специализации доктора;
* страница просмотра информации о докторах;
* страница просмотра информации о докторе;

У пользователя должна быть возможность регистрации, входа в систему или выхода из системы с любой из страниц сайта.

Главная страница необходимо для обеспечения перехода на страницы просмотра информации о медицинских организациях или докторах.

На странице просмотра информации о медицинских организациях должна отображаться краткая информация обо всех доступных медицинских учреждениях города, имеющаяся в базе данных.

Страница выбора специализации доктора должна содержать список специализаций докторов, занесенных в базу данных. На странице просмотра информации о докторах должна отображаться краткая информация о докторах выбранной специализации.

Страница просмотра информации о медицинской организации должна содержать подробную информацию о соответствующем медицинском учреждении. Также на этой странице должен быть представлен список работающих в данной медицинской организации докторов.

Страница просмотра информации о докторе должна содержать подробную информацию о соответствующем специалисте. Также на этой странице должно быть представлено расписание доктора в виде таблицы. Элементы таблицы должны представлять собой интерактивные элементы, нажимая на которые пользователь сможет записываться на прием, отменять запись на прием и т.д.

Для реализации описанных выше требований необходимо разработать веб-сервер, отвечающий на HTTP-запросы приложения, реализующего графический интерфейс пользователя. Веб-сервер должен предоставлять программный интерфейс, позволяющий получать различную информацию, находящуюся в базе данных рассматриваемой информационной системы. Также веб-сервер должен предоставлять возможность изменения данных, содержащих информацию о записях пользователей к докторам (например, для отмены записи).

## 1.3 Анализ предметной области

В рассматриваемой предметной области можно выделить следующие сущности (рисунок 2):

* Пользователь;
* Доктор;
* Организация;
* Отделение;
* Событие.

Сущность «пользователь» содержит информацию о пользователях системы, относящихся к категориям «пациент» и «доктор». Сущность доктор содержит информацию, которая относится только к пользователям с категорией «доктор». Сущность «организация» инкапсулирует информацию о медицинских учреждениях. В организацию входят отделения, информацию о которых содержит сущность «отделение». Для реализации возможности записи на прием к врачу в конкретный момент времени необходимо также выделить сущность «событие». Сущность «событие» отражает момент времени и статус этого момента времени. Статус характеризует возможность записи на прием к врачу.

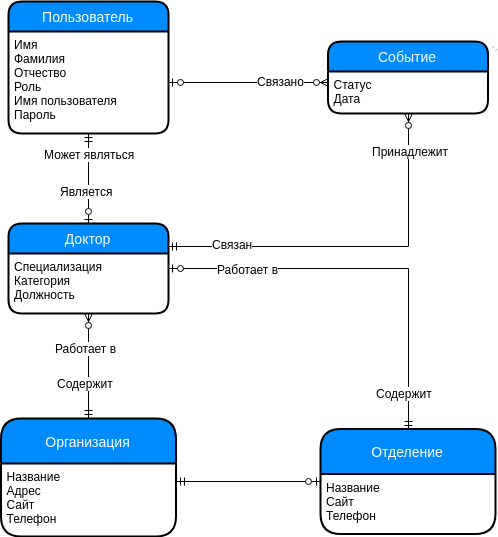


Рисунок – Концептуальная модель предметной области

Каждый зарегистрированный пользователь может являться доктором или пациентом. Доктор обязательно принадлежит какой-либо организации и какому-либо отделению этой организации. Организация может содержать несколько отделений. С каждым событием связан доктор. Событие также может быть связано с пользователем. Тогда считается, что пользователь записан на прием к доктору соответствующий этому событию момент времени.

## 1.4 Выбор технологий и средств разработки

Рассматриваемая система представляет собой веб-приложение. В общем случае его реализация может потребовать решения следующих задач:

* Разработка и реализация графического интерфейса пользователя;
* Проектирование и реализация веб-сервера;
* Проектирование структуры базы данных, а также обеспечение взаимодействия серверного программного обеспечения с сервером базы данных;

Разработка веб-интерфейсов, в настоящее время практически никогда не реализуется без использования динамического языка программирования JavaScript [2]. Данный язык программирования обладает рядом достоинств: гибкость, универсальность, популярность, большое количество готовых решений и т.д. Именно поэтому он будет использован также и для создания серверной составляющей веб-приложения. Программной платформой, позволяющей использовать JavaScript на стороне сервера является Node.js [3]. Одной из причин выбора Node.js является его популярность и развитость. Пакетная экосистема Node.js – npm, является самой большой экосистемой библиотек с открытым исходным кодом в мире.

Для построения реактивного веб-интерфейса будет использована библиотека React [24]. Реализацию веб-сервера планируется проводить с помощью веб-фреймворка Express [9], содержащего обширный набор функций мобильных и веб-приложений. При разработке системы планируется использовать в качестве базы данных PostgreSQL [10]. Причинами данного выбора являются развитость технологии, большое сообщество, а также наличие всех функциональных возможностей, которые необходимы при реализации рассматриваемой системы (функции, триггеры, правила и представления, индексы и т.д.). Для упрощения разработки и взаимодействия с сервером базы данных, планируется использовать ORM (Object-Relational Mapping). ORM позволит увеличить скорость разработки, а также абстрагироваться от нюансов работы с конкретной базой данных. В качестве ORM при разработке рассматриваемой системы планируется использовать Sequelize [11]. Очевидно, что использование любой ORM увеличивает накладные расходы, улучшая при этом поддерживаемость кода. Однако Sequelize позволяет писать и 'чистые' SQL запросы, что делает выбор данной технологии еще более обоснованным.

# 2 Проектирование программных средств и струкур данных

## 2.1 Проектирование объектной модели базы данных

В данной курсовой работе планируется использовать систему управления базами данных PostgreSQL, а для взаимодействия с ней Sequelize ORM. Sequelize позволяет определять модели, которые соответствуют таблицам базы данных. Каждая модель – это класс, а объекты данного класса представляют строку в соответствующей таблице. При определении модели необходимо указывать ее название, а также описывать все атрибуты (название, тип, ограничения и т.д.). Sequelize определяет для каждой модели статические методы, позволяющие воздействовать на соответствующую таблицу базы данных (добавлять, изменять, удалять записи, осущетсвлять выборку записей на основе различных условий и т.д.).

Выделенные в ходе анализа предметной области сущности представлены в виде объектной модели базы данных на рисунке 3.

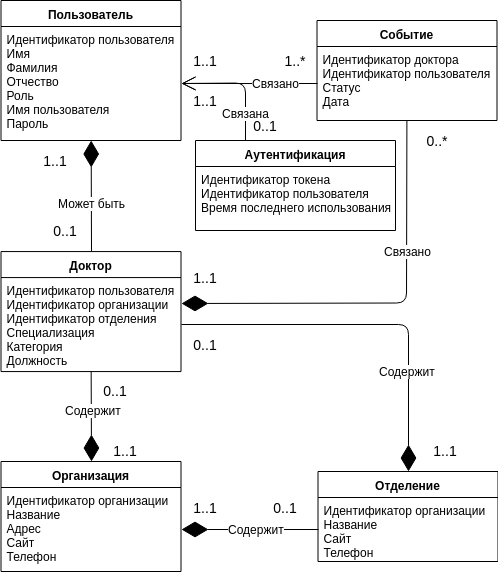


Рисунок – Объектная модель базы данных

В объектной модели базы данных предсталвен класс «Аутентификация», отвечающий за хранение токенов аутентификации.

## 2.2 Структура программного обеспечения

Рассматриваемая в данной курсовой работе система состоит из следующих основных компонентов:

* сервер (backend) – приложение, предоставляющее программный интерфейс (API) информационной системы;
* клиент (frontend) – приложение, реализующее графический интерфейс пользователя;
* сервер базы данных (PostgreSQL).

Рассмотрим подробнее структуру компонентов веб-сервера. Веб-сервер, реализованный с использованием Node.js, может состоять из следующих основных элементов (рисунок 4):

* маршрутизатор HTTP-запросов;
* сервис логирования;
* один или несколько классов ошибок;
* компоненты для работы с базой данных.

При реализации веб-сервера на Node.js обычно используется фреймворк Express, в том числе предоставляющий удобный способ маршрутизации HTTP-запросов посредством класса Router. Express также позволяет запустить сервер, прослушивающий заданный порт и принимающий HTTP-запросы для их последующей маршрутизации и обработки (класс HTTPService).

Для взаимодействия с сервером базы данных PostgreSQL планируется использовать Sequelize ORM. Для непосредственной установки соединения, а также для определения моделей (представляющих таблицы базы данных), необходимо использовать класс Sequelize.

Для удобной обработки ошибок целесообразно создать некий класс (ControlledError), наследуемый от стандартного класса ошибок JavaScript.

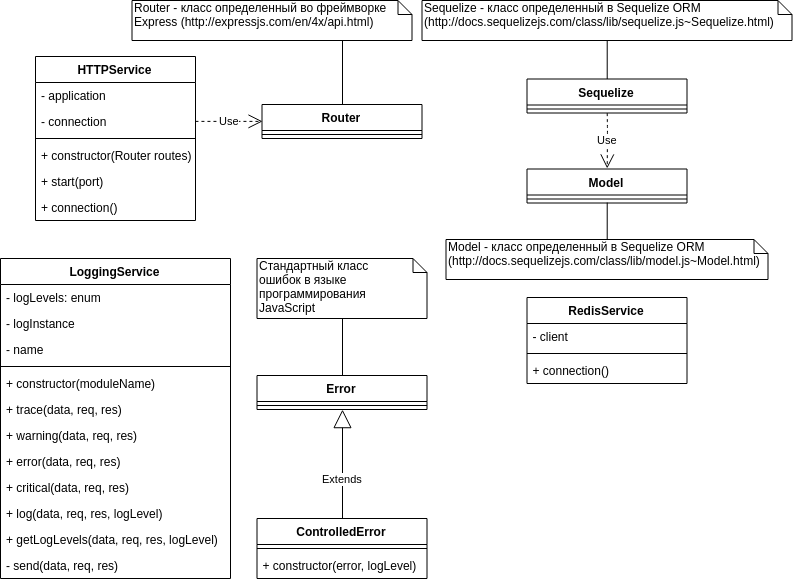


Рисунок - Диаграмма классов веб-сервера

При обработке запросов необходимо обеспечить логирование результатов. Для логирования необходимо создать класс (LoggingService), позволяющий формировать сообщения с учетом переданных параметров (наличие или отсутствие объекта ошибки, уровень сообщения), а также предоставляющий информацию о возможных уровнях логирования.

## 2.3 Модель интерфейса

В данной курсовой работе необходимо разработать программный интерфейс веб-сервера, а также графический интерфей, с помощью которого пользователи смогут осуществлять взаимодействие с информационной системой. Описание HTTP-маршрутов разрабатываемого программного интерфейса приведено в приложении А.

HTTP-запрос может включать куки (cookie) – данные, сохраненные сервером в браузере клиента. В разрабатываемой системе в куках сохраняется токен авторизации. С помощью токена авторизации программные средства, разрабатываемые в данной курсовой работе, смогут осуществлять подтверждение личности пользователя.

Модель графического интерфейса пользователя представлена в приложении Б.

# 3 Реализация программных средств

## 3.1 Компоненты системы

В данной курсовой работе разрабатываются программные средства HTTP-сервера, состоящие из следующих компонентов и подсистем (рисунок 5): package-lock.json, package.json, .babelrc, .sequelizerc, src и config. Файл package.json является основным конфигурационным файлом приложений, реализованных с использованием Node.js. В нем, помимо прочего, содержится информация о зависимостях, которые можно установить с помощью утилиты командной строки npm. Файл package-lock.json содержит информацию о версиях установленных зависимостей. Подсистема node\_modules содержит все установленные зависимости. Файл .sequelizerc является конфигурационным файлом Sequelize.

Подсистема config предназначена для хранения файлов с конфигурационными данными для подсистем src. Подсистема src содержит компоненты с исходным текстом программных средств.

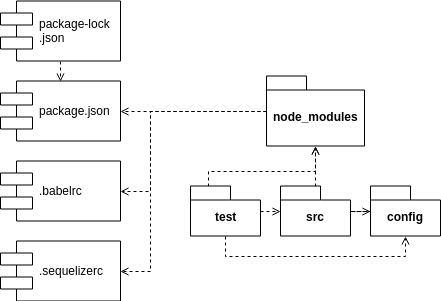


Рисунок – Диаграмма компонентов подсистемы src

Подсистема src (рисунок 6) содержит файлы исходного кода веб-сервера. Она состоит из следующих компонентов и подсистем: indes.js, http, routes, models, migrations, validators, utils, middlewares. Каждый из перечисленных компонентов и подсистем зависит от конфигурационных данных (подсистема config) и от различных сторонних npm-пакетов (подсистема node\_modules). Файл index.js является точкой входа в приложение. Он содержит исходный код, запускающий HTTP-сервер (подсистема http) и синхронизирующий все определенные модели базы данных (подсистема models).

Независимой подсистемой является подсистема migrations, в которой содержатся объекты, предназначенные для физического создания или изменения таблиц базы данных. При каждом изменении структуры базы данных необходимо добавлять в данную подсистему объект, описывающий ту или иную модификацию (добавление, удаление или изменение атрибута таблицы). Применение миграций производится с помощью утилиты командной строки sequelize.

Подсистема http предназначана для запуска веб-сервера. При реализации обработчиков HTTP-запросов были использованы функциональные возможности следующих вспомогательных подсистем:

* models;
* validators;
* utils;
* middlewares.

Подсистема models предназначена для установки соединения с сервером базы данных PosgreSQL и формировании классов, используя которые можно воздействовать на таблицы базы данных.

Подсистема validators содержит исходный код, предназначенный для валидации параметров HTTP-запросов.

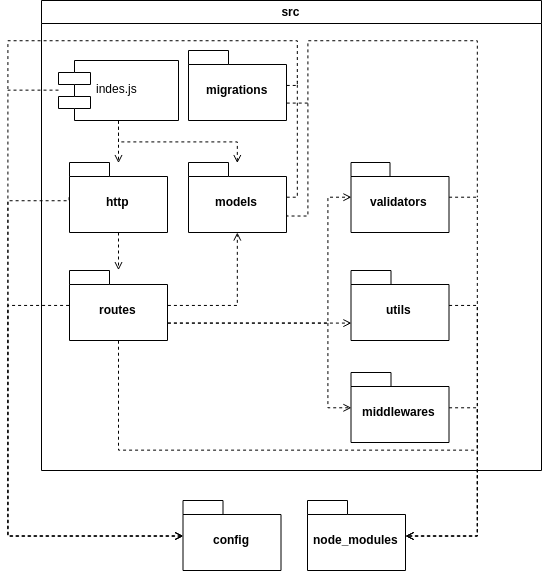


Рисунок – Диаграмма компонентов подсистемы src

Подсистема utils содержит вспомогательные компоненты, предназначенные для повторного использования при реализации обработчиков HTTP-запросов.

Подсистема middlewares содержит промежуточные обработчики, которые могут исполняться перед основными обработчиками, определенными в подсистеме routes.

## 3.2 Физическая модель базы данных

Каждому классу, разработанному в ходе проектирования объектной модели базы данных, соответствует таблица в базе данных (рисунок 7). Каждому атрибуту класса ставится в соответствие тип, доступный в Sequelize ORM. На тип атрибута могут накладываться дополнительные ограничения (область допустимых значений для чисел, возможные значения для множеств и т.д.). Для каждого атрибута указывается, может ли оно быть null-значением. Для каждой таблицы определяется первичный и внешние ключи.

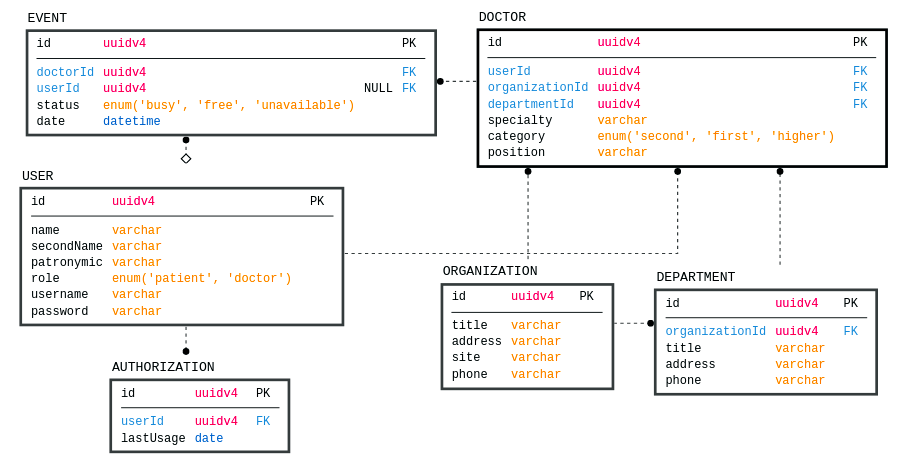


Рисунок – Физическая модель базы данных

Все первичные ключи в разрабатываемой базе данных имеют тип UUIDv4, позволяющий распределённым системам уникально идентифицировать информацию в разных подсистемах и сервисах. Для хранения времени и даты используется тип DATETIME. Для атрибутов, значения которых могут принадлежат некоторому множеству, используется тип данных ENUM. Строковые значения определяются типом VARCHAR.

# 4 Тестирование программных средств

Тестирование – этап жизненного цикла разработки программного обеспечения, позволяющий сравнить реальное и ожидаемое поведение системы. По объектам тестирования выделяют:

* функциональное тестирование;
* тестирование производительности;
* конфигурационное тестирование;
* юзабилити-тестирование;
* тестирование интерфейса пользователя.

Объектом тестирования разработанных программных средств является функциональная составляющая. То есть необходимо провести функциональное тестирование и проверить способность системы обрабатывать HTTP-запросы в различных условиях.

Разработанные функциональные тесты, позволяющие судить о корректности реализованных функциональных возможностей, представлены в таблице 1.

Таблица 1 – Функциональные тесты разрабатываемого веб-приложения

| **Вариант использования** | **Описание теста** | **Ожидаемый результата** | **Полученный результат** | **Результат теста** |
| --- | --- | --- | --- | --- |
| Выход из системы | Пользователь нажал кнопку выйти | Сервер удалил токен аутентификации, в браузере удалены cookie | Сервер удалил токен аутентификации, в браузере удалены cookie | Тест пройден |
| Аутентифицироваться | Пользователь ввел имя пользователя и пароль в поля ввода и нажал кнопку войти | Сервер создал токен доступа и сохранил его в браузере клиента | Сервер создал токен доступа и сохранил его в браузере клиента | Тест пройден |
| Зарегистрироваться | Пользователь заполнил поля ввода и нажал кнопку | Уведомление об успешнм создании учетной записи | Уведомление об успешнм создании учетной записи | Тест пройден |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вариант использования** | **Описание теста** | **Ожидаемый результата** | **Полученный результат** | **Результат теста** |
|  | «зарегистрироваться» |  |  |  |
| Редактировать данные аккаунта | Пользователь вводит новые значения в поля ввода формы обновления и нажимает кнопку «обновить» | Уведомление об успешнм обновлении учетной записи | Уведомление об успешнм обновлении учетной записи | Тест пройден |
| Просмотреть информацию о врачах | Пользователь переходит на страницу «Расписание/Аллерголог» | Отображается список аллергологов | Отображается список аллергологов | Тест пройден |
| Просмотреть информацию о медицинских организациях | Пользователь переходит на страницу «Медицинские организации» | Отображается список организация из отделений | Отображается список организация из отделений | Тест пройден |
| Записаться к врачу | Пользователь пытается записаться к врачу | Время, на которое осуществлялась запись становится занятым | Время, на которое осуществлялась запись становится занятым | Тест пройден |
| Отменить запись к врачу | Пользователь пытается отменить запись к врачу | Время, на которое осуществлялась запись становится свободным | Время, на которое осуществлялась запись становится свободным | Тест пройден |
| Запретить запись на определенное время | Доктор пытается запретить запись на определенное время | Время становится недоступным для записи | Время становится недоступным для записи | Тест пройден |

Тестирование показало корректность разработанных программных средств, а также их соответствие предъявленным функциональным требованиям.

# Заключение

В ходе выполнения данной курсовой работы было реализовано приложение записи на прием к врачу для информационной системы медицинских организаций города. При разработке был спроектирован программный интерфейс, позволяющий воздействовать на рассматриваемую подсистему с помощью HTTP-запросов. С учетом представленных требований был произведен анализ предметной области. На его основе были спроектированы необходимые структуры данных (структура реляционной базы данных PostgreSQL). Для реализации HTTP-сервера и его взаимодействия с сервером базы данных использовались классы сторонних библиотек (Router, Sequelize, Model) и классы, спроектированные и реализованные на их основе (HTTPService, LoggingService, RedisService, ControlledError). Реализация графического интферфейса пользователя проводилась с использованием библиотеки React. Это позволило создать качественное и быстрореагирующее на управляющие воздействия пользователя клиентское приложение. Проверка качества кода осуществлена с помощью функционального тестирования. Результаты тестирования показывают корректную работу приложения.

# Список использованных источников

1. Жуйков Р. Методы предварительной оптимизации программ на языке JavaScript / Жуйков Р., Шарыгин Е. // Труды института системного программирования РАН. - 2015. - №6. - с. 67 - 86.
2. Официальный сайт Node.js [Электронный ресурс]. - URL: https://nodejs.org/ (дата обращения: 08.03.19).
3. Официальный сайт React.js [Электронный ресурс]. - URL: https://reactjs.org/ (дата обращения: 15.03.19).
4. Официальный сайт Vue.js [Электронный ресурс]. - URL: https://vuejs.org/ (дата обращения: 20.03.19).
5. Официальный сайт Express.js [Электронный ресурс]. - URL: https://expressjs.com/ (дата обращения: 21.03.19).
6. Официальный сайт PostgreSQL [Электронный ресурс]. - URL: https://www.postgresql.org/ (дата обращения: 25.03.19).
7. Официальный сайт Sequelize [Электронный ресурс]. - URL: http://docs.sequelizejs.com/ (дата обращения: 26.03.19).

**МОДЕЛЬ ПРОГРАММНОГО ИНТЕРФЕЙСА**

Приложение A

(обязательное)

Таблица А.1 – Модель программного интерфейса

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Группа | Тип | Маршрут | Авторизация \* | Назначение | Параметры | | | Пользователь | Описание действий |
| Имя | Тип | Комментарий |
| Врачи | GET | /doctors | Публичный | Получение списка врачей | specialty | query | Параметр указывает специальность врачей (необязательный) | Все | Необходимо вернуть список всех врачей. Если указан параметр specialty, то необходимо вернуть только врачей с указанной специальностью. Если указан параметр organization, то необходимо вернуть только врачей из указанной организации. Необходимо вернуть массив следующего вида: {  <doctor identifier>: {  "userId": "...",  "name": "...",  "secondName": "...",  "patronymic": "...",  "organizationId": "...",  "departmentId": "...",  "specialty": "...",  "category": "...",  "position": "..."  }  ... } |
| organizationId | query | Параметр указыват идентификатор организации, к которой должны принадлежать врачи (необязательный) |
| doctorsIds | query | Параметр указывает список идентификаторов, которые должны быть возвращены (необязательный) |
| GET | /doctors/:id | Публичный | Получение полной информации о враче | id | params | Параметр указывает идентификатор врача, о котором требуется вернуть информацию | Все | Необходимо вернуть информацию о преподавателе в следующем виде: {  <doctor identifier>: {  "userId": "...",  "name": "...",  "secondName": "...",  "patronymic": "...",  "organizationId": "...",  "departmentId": "...",  "specialty": "...",  "category": "...",  "position": "..."  } } |
| GET | /doctors/specialties | Публичный | Получение списка специальностей врачей |  |  |  | Все | Необходимо вернуть информацию в следующем виде: {[  "Специальность 1",  "Специальность 2",  "Специальность 3",  ... ]} |
| Организации | GET | /organizations | Публичный | Получение списка организаций |  |  |  | Все | Необходимо вернуть информацию об организациях в следующем виде: {  <organization identifier>: {  "title": "...",  "address": "..."  } } |
| GET | /organizations/:id | Публичный | Получение полной информации об организации | id | params | Параметр указывает идентификатор организации, о которой требуется вернуть информацию | Все | Необходимо вернуть информацию об организации в следующем виде: {  <organization identifier>: {  "title": "...",  "address": "...",  "site": "...",  "phone": "..."  } } |
| Отделения | GET | /departments | Публичный | Получение списка отделений | organizationId | query | Параметр указывает идентификатор организации, к которой должны принадлежать возвращаемые отделения (необязательный) | Все | Необходимо вернуть список всех отделений. Если указан параметр organizationId, то необходимо вернуть только отделения с указанным идентификатором организации. Информацию необходимо вернуть в следующем виде: {  <department identifier>: {  "organizationId": "...",  "title": "...",  "address": "...",  "phone": "..."  },  ... } |
| Пользователи | GET | /users/self | Токен | Получение полной информации о пользователе |  |  |  | Все | Необходимо вернуть информацию об пользователе в следующем виде: {  "id": "...",  "name": "...",  "secondName": "...",  "patronymic": "...",  "role": "...",  "username": "...",  "doctorId": "..." } |
| GET | /users | Публичный | Получение краткой информации о пользователях | userIds | query | Массив идентификаторов пользователей, о которых требуется получить информацию | Все | Необходимо вернуть информацию об пользователе в следующем виде: {  "name": "...",  "secondName": "...",  "patronymic": "..." } |
| POST | /users/signin | Публичный | Аутентификация пользователя | username | body | Имя пользователя | Все | Создать токен и сохранить его в куках браузера |
| password | body | Пароль |
| POST | /users/signout | Токен | Выход пользователя из системы |  |  |  | Все | Удалить токен и куков браузера |
| POST | /users/signup | Публичный | Регистрация пользователя в системе | name | body | Имя | Все |  |
| secondName | body | Фамилия |
| patronymic | body | Отчество |
| username | body | Имя пользователя |
| password | body | Пароль |
| PATCH | /users/self | Токен | Обновление информации о пользователе | name | body | Имя (необязательный) | Все | Обновление информации о пользователе. Обновлять необходимо только те поля, которые указаны в параметрах. Необходимо вернуть информацию об обновленном пользователе в следующем виде: {  <user identifier>: {  "name": "...",  "secondName": "...",  "patronymic": "...",  "role": "...",  "username": "..."  } } |
| secondName | body | Фамилия (необязательный) |
| patronymic | body | Отчество (необязательный) |
| username | body | Имя пользователя (необязательный) |
| password | body | Пароль (необязательный) |
| События | GET | /events | Публичный | Получение списка событий | doctorId | query | Параметр указыват идентификатор врача, чье расписание необходимо вернуть | Все | Необходимо вернуть информацию только о будущих событиях (поле date должно быть больше Date.now()) Необходимо вернуть информацию в следующем виде: {  <event identifier>: {  "status": "...",  "date": "...",  "userId": "...",  "doctorId": "..."  },  ... } |
| PATCH | /events/:id | Токен | Обновление информации о событии | id | params | Идентификатор события, о котором необходимо обновить информацию | Все | Необходимо проверить, что происходит попытка изменения будущего события (поле date должно быть больше Date.now()). Если это не так, необходимо вернуть ошибку EVENT\_NOT\_FOUND. Запрос могут делать пациенты или доктора. Если значение параметра status = 'busy', то:  если запрос сделал доктор, то проверить, что он пытаестя занять не свое собственное событие  проверить, что статус изменяемого события равен 'free'. Если значение параметра равно 'free', то:  если запрос сделал пациент, проверить, что он отменяет связанную с ним запись  если запрос сделал доктор, проверить, что он меняет свое событие Если значение параметра равно 'unavailable', то:  проверить, что запрос сделал доктор и что событие, которое он меняет его Необходимо вернуть обновленное событие в следующем виде: {  <event identifier>: {  "status": "...",  "date": "...",  "userId": "...",  "doctorId": "..."  } } |
| status | body | Новый статус события. |

**МОДЕЛЬ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА**

Приложение Б

(обязательное)

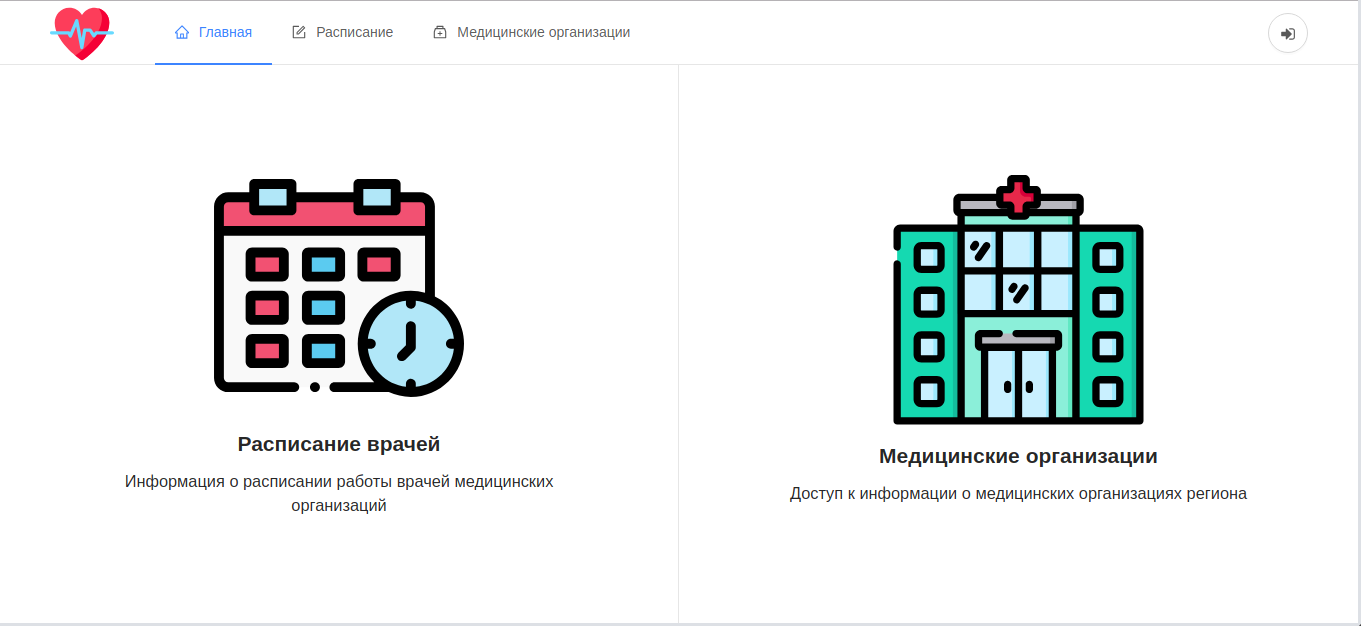


Рисунок Б. – Главная страница

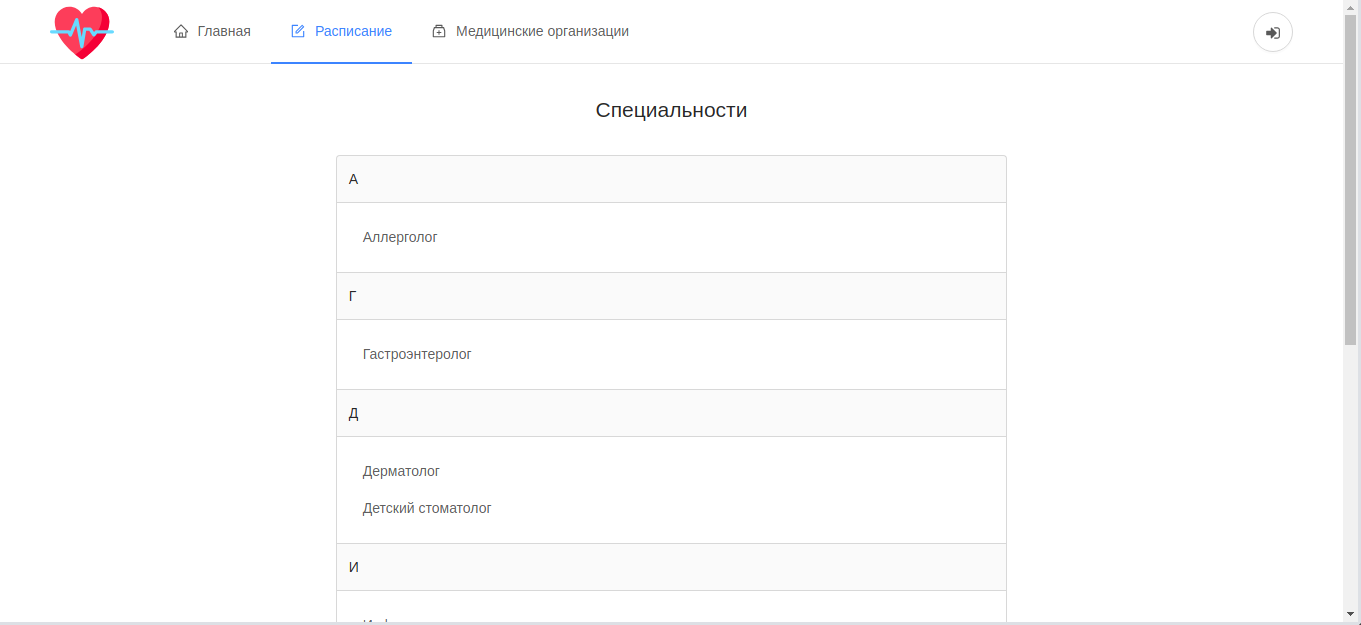


Рисунок Б.2 – Страница выбора специализации доктора

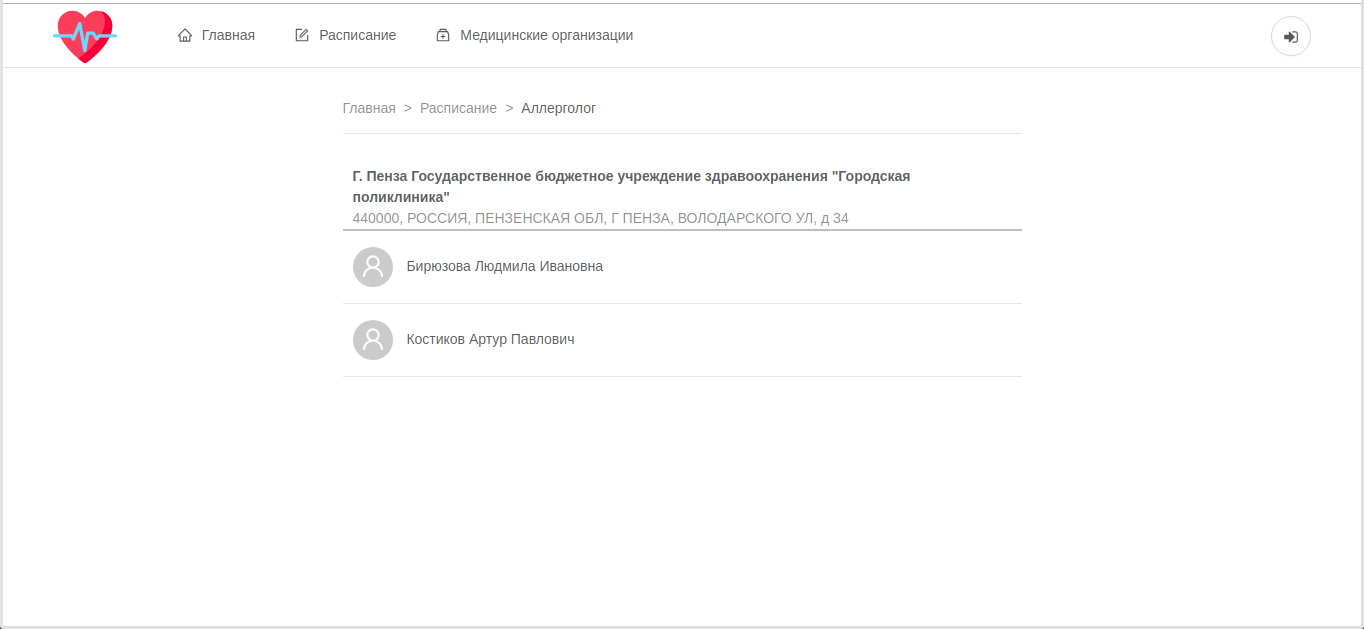


Рисунок Б.3 – Страница просмотра информации о докторах

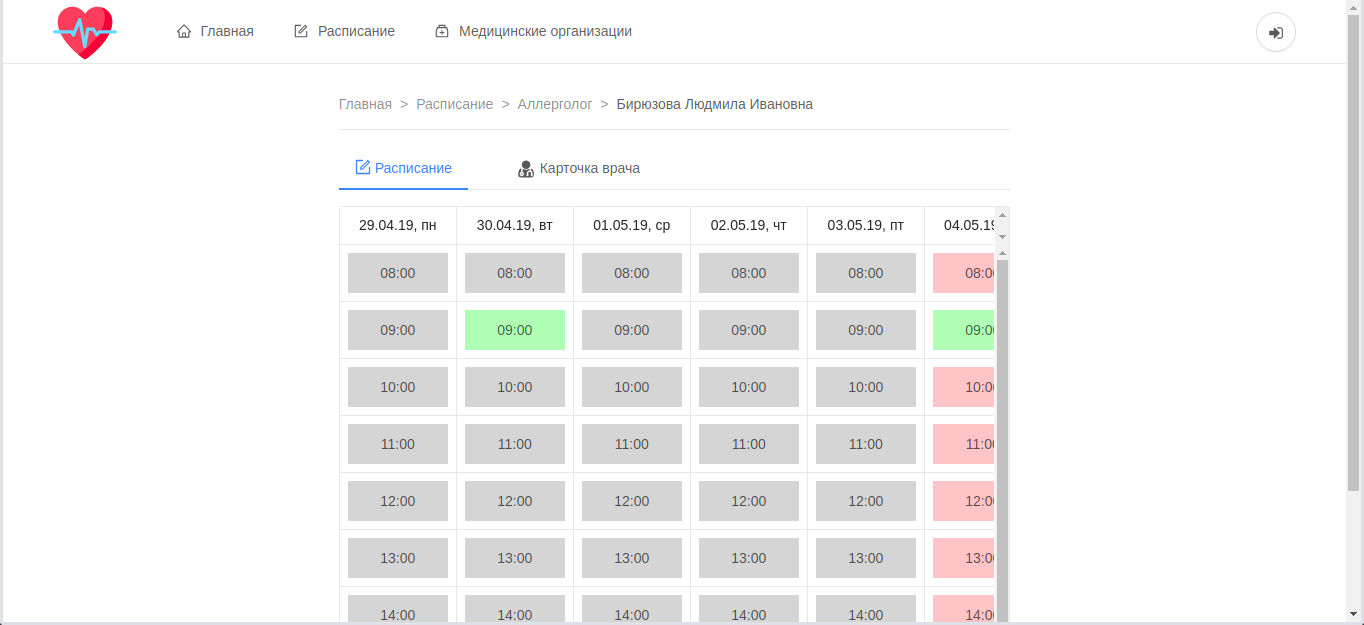


Рисунок Б.4 – Страница просмотра информации о докторе (расписание)

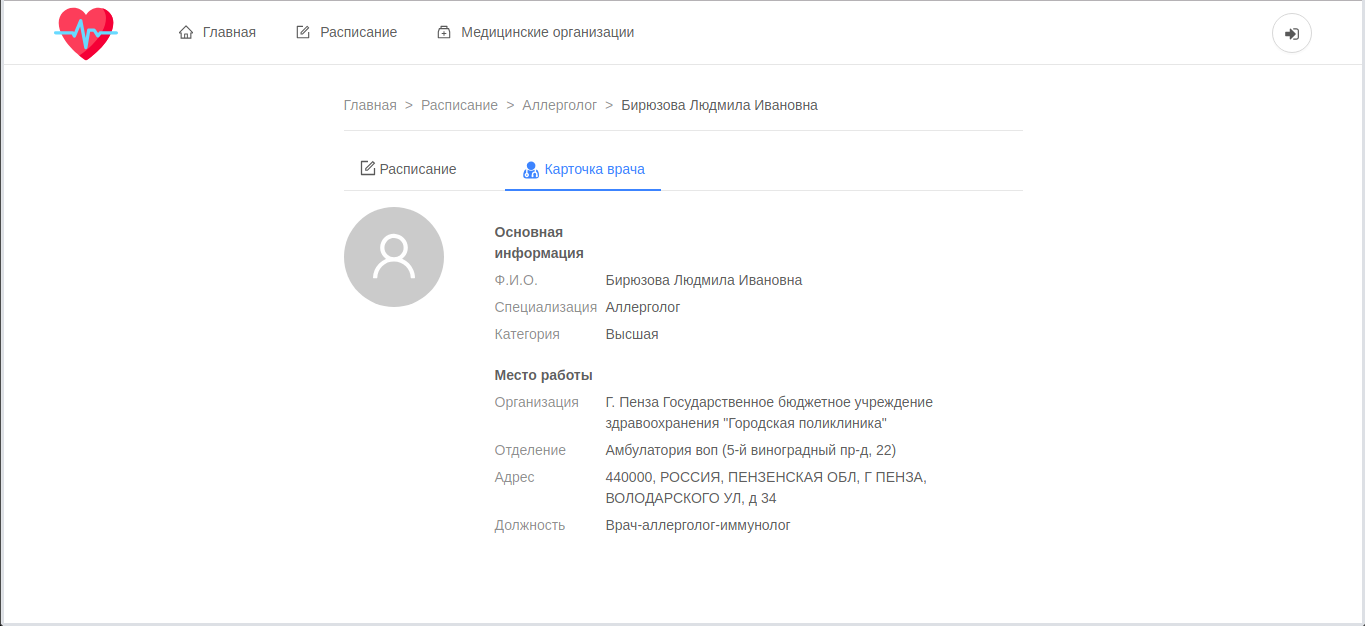


Рисунок Б.5 – Страница просмотра информации о докторе (карточка врача)

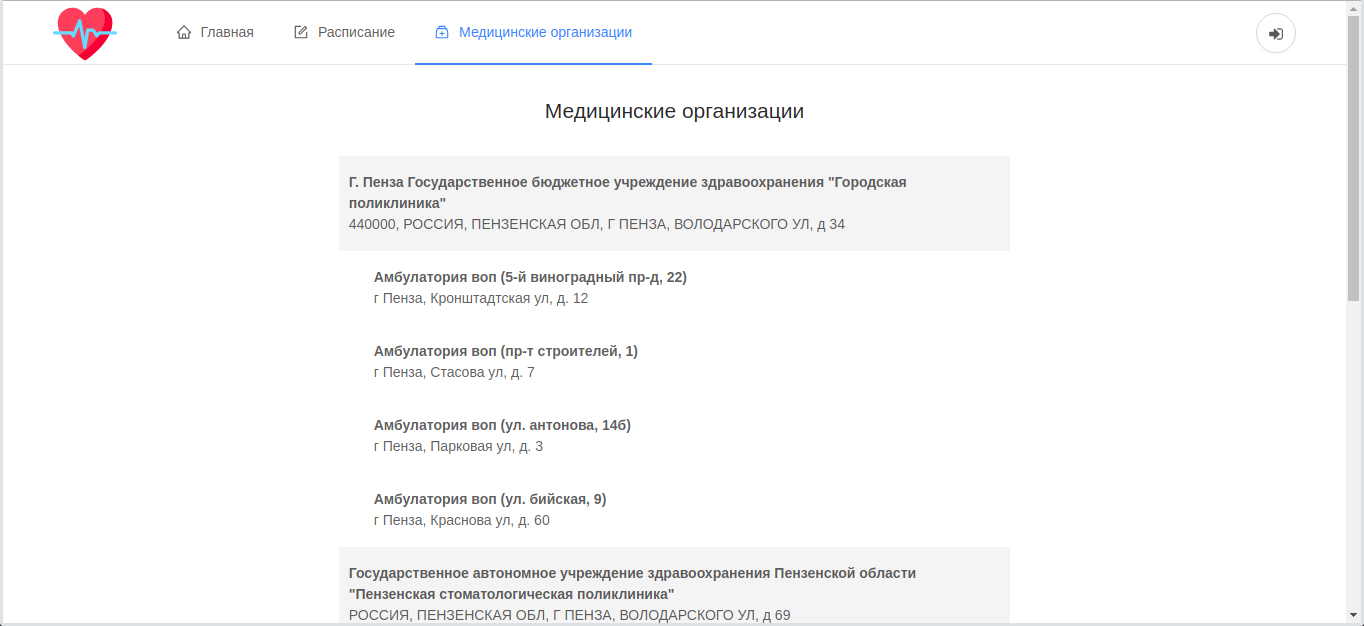


Рисунок Б.6 – Страница просмотра информации о медицинских организациях

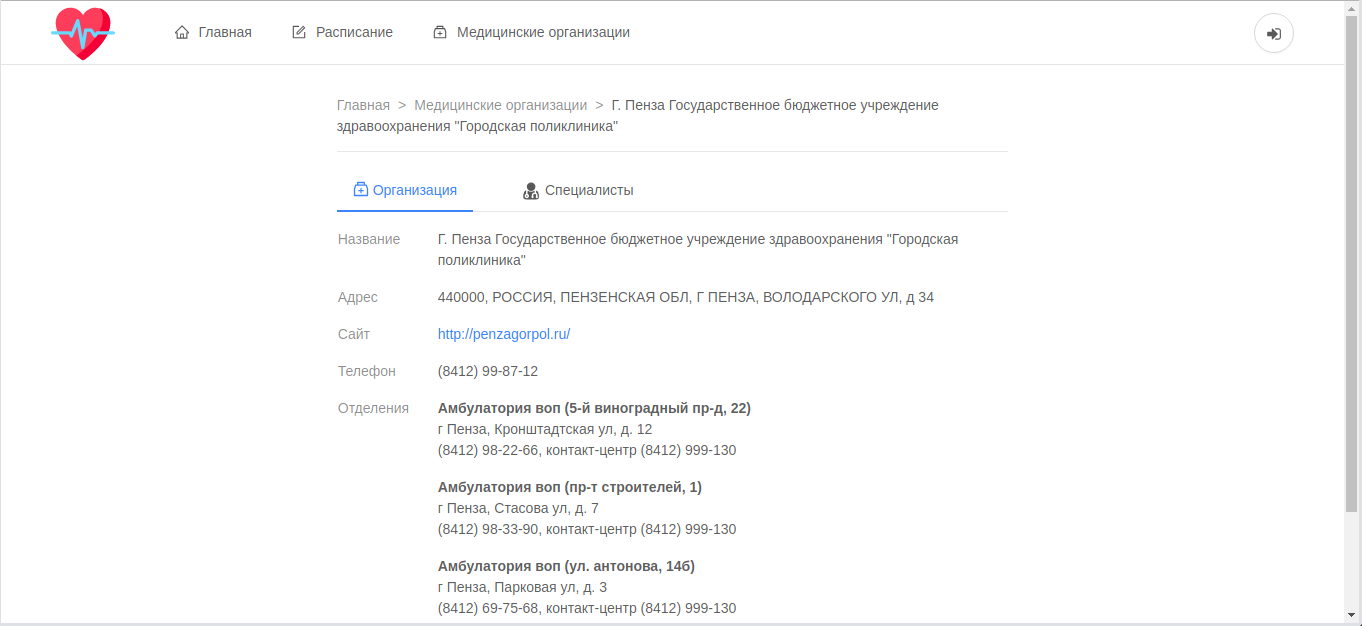


Рисунок Б.7 – Страница просмотра информации о медицинской организациии

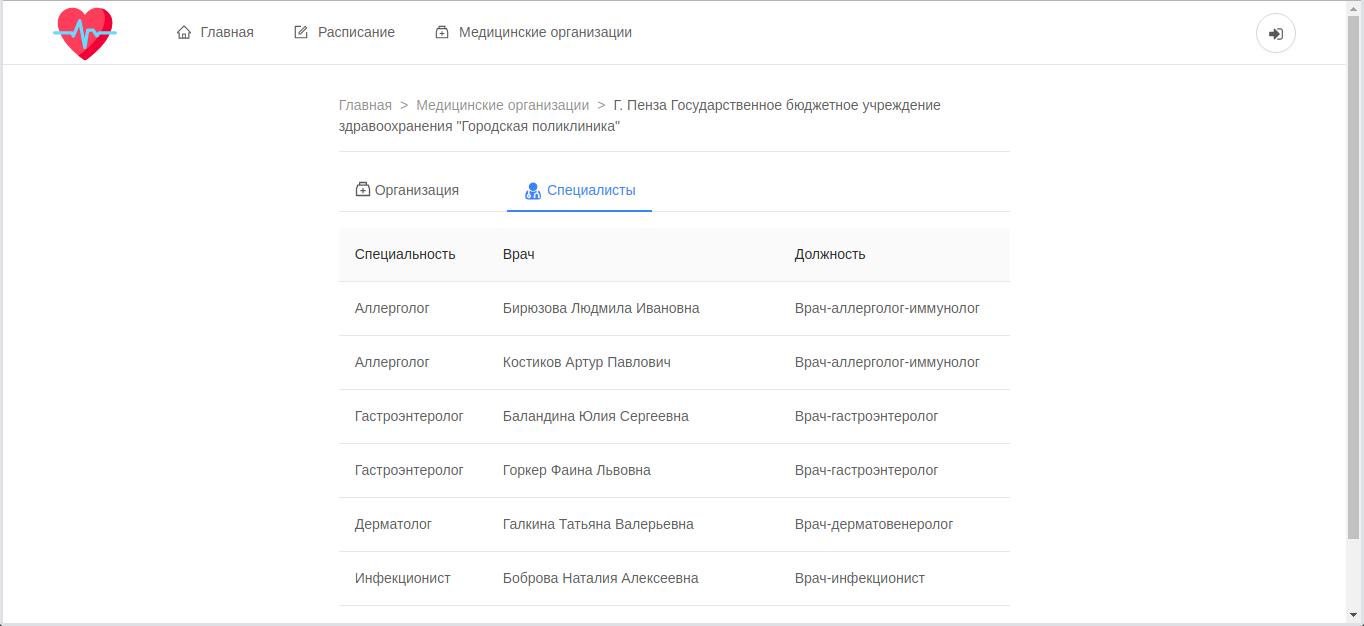


Рисунок Б.8 – Страница просмотра информации о специалистах медицинской организации