

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»

Магистерская программа: «Прикладной искусственный интеллект»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

**РАЗРАБОТКА БАЗЫ ЗНАНИЙ ДЛЯ ИНТЕЛЛЕКТУАЛЬНОЙ
СИСТЕМЫ ПОДДЕРЖКИ ВРАЧА-РЕФЛЕКСОТЕРАПЕВТА**

Студент Касимов 27.05.2024 Касимовская Алёна Дмитриевна
(подпись, дата) (ФИО полностью)

Руководитель Бождай 27.05.2024 Бождай А. С.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр Бычков 27.05.2024 Бычков А. С.
(подпись, дата) (фамилия, инициалы)

Рецензент Мартышкин 10.06.2024 Мартышкин А. И., зав. каф. Программирования ПензГУ
(подпись, дата) (фамилия, инициалы) (должность, место работы)

Работа допущена к защите (протокол зас. кафедры от 27.05.2024 № 11)

Заведующий кафедрой Митрохин Митрохин М. А.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 14.06.24 № 17)

Секретарь ГЭК Гурин Гурин Е. И.
(подпись) (фамилия, инициалы)

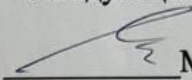
Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой

 М. А. Митрохин
«13» сентября 2022 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ МАГИСТРА

1. Студент Касимовская Алёна Дмитриевна гр. 22ВВИм2 факультета вычислительной техники, направления 09.04.01.

2. Тема работы «Разработка базы знаний для интеллектуальной системы поддержки врача-рефлексотерапевта».

Тема утверждена приказом ПГУ № 765/о от "13" сентября 2022 г.

3. Руководитель магистерской диссертации д.т.н., профессор кафедры САПР ПГУ Бождай А. С.

4. Задание на работу (предмет и назначение разработки, функциональные, программные, технические требования и т.п.)

Предмет работы: разработка инфологической, логической и физической модели предметной области, связанной со структурой человеческого организма с точки зрения традиционной китайской медицины.

Назначение разработки: повышение эффективности работы и оперативной информационной поддержки врача-рефлексотерапевта.

Функциональные требования: наличие графического интерфейса; структурированная подача данных; возможность принимать запросы от пользователей; отображение результата запроса.

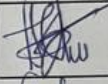
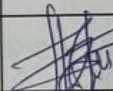
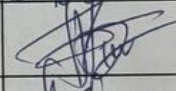
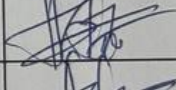
Программные требования: Microsoft Visual Studio Code, язык программирования JavaScript.

Технические требования: персональный компьютер, ОС Microsoft Windows 10.

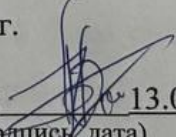
5. Перечень подлежащих разработке вопросов:

- 1) Анализ технического задания.
- 2) Анализ предметной области.
- 3) Разработка базы знаний.
- 4) Разработка программного обеспечения.

6. Календарный график выполнения работы

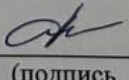
№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучить теоретические основы	10%	27.09.2022-15.11.2023	
2.	Анализ технических заданий, анализ существующих информационных систем в области китайской медицины	10%	16.11.2023-22.03.2023	
3.	Выбор инструментов для разработки и утверждение функционала системы	5%	23.03.2023-14.06.2023	
4.	Спроектировать базу данных	10%	15.06.2023-10.09.2023	
5.	Написать логику работы приложения и разработать программное обеспечение	55%	11.09.2023-17.04.2024	
6.	Апробация разработанного клиент-серверного приложения в форме тестирования	5%	18.04.2024-9.05.2024	
7.	Оформление выпускной работы	5%	10.05.2024-28.05.2024	

Дата выдачи задания "13" сентября 2022 г.

Руководитель магистерской диссертации  13.09.2022 Бождай А.С.
(подпись, дата)

Задание к исполнению принял студент  13.09.2022 Касимовская А. Д.
(подпись, дата)

Работу к защите допустить
Декан факультета

 Фионова Л.Р.
(подпись, дата)
11.06.2024

РЕФЕРАТ

Магистерская работа содержит ~~133~~ листа, 50 рисунков, 15 источников и 2 приложения

БАЗА ЗНАНИЙ, ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА, СЛУЧАЙНЫЙ
ЛЕС, ИНФОРМАЦИОННАЯ ПОДДЕРЖКА РЕФЛЕКСОТЕРАПЕВТА,
КИТАЙСКАЯ МЕДИЦИНА, JAVA SCRIPT, REACT.

Цель работы – повышение эффективности работы врача-рефлексотерапевта за счет разработки базы знаний предметной области с последующим ее использованием в структуре интеллектуальной системы.

Необходимыми задачами для достижения цели являются: анализ технического задания, исследование предметной области, анализ существующих решений, формализованное представление предметной области за счет создания инфологической, логической и физической моделей базы знаний, программная реализация базы знаний для интеллектуальной системы поддержки врача-рефлексотерапевта на основе созданных моделей, программное обеспечение для управления базой знаний, организации системы запросов и визуализации информационного контента.

Объектом разработки является база знаний для интеллектуальной системы поддержки врача рефлексотерапевта.

В процессе работы были выполнены все поставленные задачи, исследованы существующие решения. На основе проведенной работы была предложена реализация базы знаний для интеллектуальной системы врача-рефлексотерапевта, на основе которой было разработано демонстрационное программное обеспечение в среде разработки Visual Studio Code на языке Java Script.

ПГУ 1.090401.32.001 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата
Разраб.		Касимовская А.Д.		27.05.20
Провер.		Бождай А.С.		27.05.20
Н. Контр		Бычков А.С.		27.05.20
Утверд.		Митрохин М.А.		27.05.20
Разработка базы знаний для интеллектуальной системы поддержки врача-рефлексотерапевта Пояснительная записка				
		Лит.	Лист	Листов
			4	133
ФВТ, каф. ВТ гр. 22ВВим2				

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ	6
ВВЕДЕНИЕ	7
1 Анализ технического задания	10
1.1 Анализ функциональных требований	10
1.2 Анализ нефункциональных требований	10
1.3 Используемые технологии и средства разработки	11
2 Анализ предметной области	12
2.1 Введение в предметную область	12
2.2 Исследование предметной области	16
2.3 Обзор технических требований к интеллектуальной системе поддержки врача-рефлексотерапевта	19
2.4 Обзор аналогов	23
3 Разработка базы знаний	28
3.1 Разработка программного обеспечения для реализации запросов к базе знаний	31
3.2 Алгоритм функционирования системы	37
4 Разработка интерфейса пользователя и сценарии работы системы	44
ЗАКЛЮЧЕНИЕ	59
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	61
ПРИЛОЖЕНИЕ А Листинг программы	63
ПРИЛОЖЕНИЕ Б Презентация	124

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей магистерской диссертации применяются следующие сокращения и обозначения

БД – База данных

БЗ – База знаний

ИС – Интеллектуальная система

КНР – Китайская Народная Республика

ОС – Операционная система

СУБД – Система управления базами данных

ACID (Atomicity, Consistency, Isolation, Durability) – требования к транзакционной системе, обеспечивающие наиболее надёжную и предсказуемую её работу

API (Application Programming Interface) – описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой

HTTP (Hyper Text Transfer Protocol) – протокол передачи гипертекста

ORM (Object-Relational Mapping) – объектно-реляционное отображение

SQL (Structured Query Language) – язык структурированных запросов

ВВЕДЕНИЕ

В последнее время, набирают популярность интеллектуальные и экспертные системы в области медицины различных направлений. Особую роль при этом имеет проектирование базы знаний в полной мере отображающей специфику предметной области. Востребованность систем с структурированной формализацией процессов принятия решений растет при изучении новых областей. Так, например, китайская медицина набирает популярность в развитых странах, многие люди интересуются альтернативными методами лечения и подходами к заботе о своем здоровье. Но на данный момент за пределами КНР, из-за раннего предпочтения медикаментозному лечению, существует очень маленькое количество аналогов в данной сфере и многие из них ограничены в своём доступе. Поэтому в данной нише будет востребована разработка и усовершенствование различных информационных систем. Приложение, предоставляющее информацию, может быть полезным как для профессионала в сфере китайской медицины, так и для любого человека в качестве обучающего источника. Программный продукт сможет быть инструментом для специалиста, который поможет обеспечить более качественное обслуживание пациентов, ускоряя выбор оптимального лечения.

Рефлексотерапевт – это специалист, занимающийся диагностикой, лечением и профилактикой различных заболеваний за счет стимуляции биологически активных точек. Используемые методы специалистом разнообразны, но работают по одному принципу логики: воздействие на конкретные точки на теле, где каждая точка связана с конкретным органом. Или можно сказать иначе, что точки на теле отражают состояние органов и систем внутри организма и с помощью стимуляции точек рефлексотерапии пытаются восстановить баланс энергии в меридианах и улучшить функцию соответствующих органов и систем.

Интеллектуальная система – это техническая или программная система, способная решать задачи, принадлежащие конкретной предметной области, знания о которой хранятся в памяти такой системы. Структура ИС включает три основных блока – базу знаний, механизм вывода решений и интеллектуальный интерфейс. База знаний – БД, содержащая правила вывода и информацию о человеческом опыте и знаниях в некоторой предметной области. Главным преимуществом базы знаний это экономия времени при поиске нужной информации.

Объектом магистерской работы являются база знаний для интеллектуальной системы поддержки врача рефлексотерапевта.

Предметом магистерской работы является разработка инфологической, логической и физической модели предметной области, связанной со структурой человеческого организма с точки зрения традиционной китайской медицины.

Целью данной магистерской диссертации является повышение эффективности работы и оперативной информационной поддержки врача-рефлексотерапевта.

Для достижения поставленной цели необходимо выполнять ряд задач, а именно:

- провести анализ технического задания,
- провести исследование предметной области,
- произвести обзор и анализ существующих решений,
- выполнить формализованное представление предметной области за счет создания инфологической, логической и физической моделей базы знаний,
- реализовать базу знаний для интеллектуальной системы поддержки врача-рефлексотерапевта на основе созданных моделей,
- разработать программное обеспечение для управления базой знаний, организации системы запросов и визуализации информационного контента.

В соответствии с поставленными задачами можно выделить четыре основных раздела данной магистерской работы.

В первом разделе происходит анализ технического задания, описывается выбор средств разработки с учётом, описанных функциональных, программных и технических требований в соответствии заданием магистерской работы.

В втором разделе предоставлено пояснение основных принципов традиционного китайского подхода к медицине и особенностей работы врача-рефлексотерапевта с последующим исследованием и анализом таких аспектов как методы воздействия, расположение и функциональность точек. Помимо были рассмотрены осуществленные способы разработки информационной системы. В конце раздела представлен обзор уже существующих решений и проведена сравнительная характеристика.

В третьем разделе описана разработка базы знаний.

В четвёртом разделе находится описание разработки интерфейса пользователя и ее интеграции.

В результате выполнения всех поставленных задач предложена формализация предметной области с последующей программной реализацией базы знаний в составе интеллектуальной системы поддержки врача-рефлексотерапевта.

1 Анализ технического задания

В рамках анализа технического задания необходимо установить функциональные, программные и технические требования к предстоящей разработке.

1.1 Анализ функциональных требований

Назначение разрабатываемого программного обеспечения – информационная поддержка врача-рефлексотерапевта за счет разработки и предоставления ему базы знаний предметной области, организации системы запросов и визуализации информационного контента. Программное обеспечение должно соответствовать следующим требованиям:

- иметь пользовательский интерфейс,
- иметь формализованное представление предметной области,
- иметь поисковую строку, где при вводе текста обновляется список результатов, отображая только элементы, соответствующие введенному запросу,
- иметь возможность взаимодействие с элементами списка.

1.2 Анализ нефункциональных требований

Нефункциональные требования, которые должна соблюдать и демонстрировать система:

- надежность. Приложение должно быть стабильным и надежным, минимизируя возможность сбоев,
- удобство использования. Интерфейс пользователя должен быть интуитивным для легко взаимодействия с приложением и простым в использовании,
- производительность кода. Код должен быть чистым, структурированным для обеспечения легкости поддержки и расширения.

1.3 Используемые технологии и средства разработки

Система управления базами данных была выбрана PostgreSQL. Это мощная объектно-реляционная СУБД. Преимущества PostgreSQL: надежное хранение, масштабируемость, обработка и управление данными.

Код для создания пользовательского интерфейса был написан на JavaScript с использованием библиотек React.

JavaScript – это интерпретируемый язык программирования, который широко используется для создания динамических интерактивных веб-страниц. React – это библиотека JavaScript которая используется для создания пользовательских интерфейсов веб-приложений.

Серверная часть выполнена с помощью Node.js – это среда выполнения JavaScript на северной стороне.

В качестве редактора кода используется Visual Studio Code. Преимущественно выбран из-за поддержания выбранного языка программирования, удобство использования и бесплатной эксплуатации.

2 Анализ предметной области

2.1 Введение в предметную область

Компьютерные интеллектуальные и экспертные системы медицинского назначения распространены в современном здравоохранении. Эти системы используют для улучшения диагностики, лечения и управления медицинскими данными. Они существенно повышают точность и эффективность медицинских услуг, снижают ошибки и облегчают работу медицинского персонала.

Основные направления использования интеллектуальных и экспертных систем:

- диагностика заболеваний. Системы поддержки принятия решений: эти системы помогают врачам в постановке диагнозов, анализируя медицинские данные пациента и сопоставляя их с большими базами данных, содержащими информацию о симптомах, результатах анализов и исторических данных по заболеваниям,
- управление медицинскими данными. Электронные медицинские записи: интеллектуальные системы управления медицинскими данными, позволяют эффективно собирать, хранить и анализировать данные пациентов, обеспечивая доступ к ним для всех участников медицинского процесса,
- увеличение доступа к медицинским услугам. Дистанционные консультации: позволяют пациентам получать медицинские консультации через видеозвонки, уменьшая необходимость в личных визитах к врачу и облегчая доступ к медицинской помощи для людей в удаленных районах,
- мониторинг хронических заболеваний. Носимые устройства и удаленные сенсоры: используются для постоянного мониторинга состояния здоровья пациентов с хроническими заболеваниями, такими как диабет и сердечные болезни, передавая данные врачам в реальном времени.

Перспективы развития таких систем включают улучшение алгоритмов машинного обучения, интеграцию с новыми источниками данных, а также расширение использования телемедицины и дистанционного мониторинга.

Традиционная китайская медицина имеет большую популярность как в Китае, так и за его пределами и хорошо используется параллельно с классической медициной. Западная медицина часто опирается на внешние вмешательства, а китайская медицина напротив, стимулирует внутренние силы оздоровления, присущие каждому человеку. Особенностью традиционной китайской медицины является то, что ее методы лечения составляют единую целую систему из теории и методах. Фундаментальная основа для лечения является принцип цельности. Китайская медицина рассматривает здоровье и болезнь через призму жизненной энергии, которая течет через определенные пути, называемые меридианами, обеспечивая выполнение функций тела. Когда жизненная энергия течет свободно, гармонично и в достаточном количестве, то это приводит к хорошему здоровью, в организме порядок и гармония. Если свободное течение будет нарушено блокировками в потоке и дисбалансом, то организм заболеет, так как возникнут беспорядок и дисгармония, и это выразится в недомогании.

Основной целью лечения является восстановление и поддержания здоровья, ранняя диагностика заболеваний, содействие общему благополучию и сбалансированного тонуса организма. Рефлекторные терапевты, опираясь на опыт многих тысячелетий, знают как должен действовать для достижения результата. Но также он может назначать лечение непосредственно на основе теории, из которой он выведет точный результат обследования. Теория и методы исцеления образуют очень логичную и цельную медицинскую систему [2].

Процесс работы врача-рефлексотерапевта начинается с осмотра и опроса. Врач выслушивает жалобы, задает вопросы насчет общего состояния здоровья и уточняет особенности симптомов. Составив предварительное мнение, доктор приступает к осмотру.

Диагностика заключается в «прощупывании» рефлексогенных зон, отвечающих за работу больного органа. Надавливая на определенные области, врач наблюдает за реакцией и делает вывод о его состоянии.

С учетом полученных данных определяется наиболее подходящий метод лечения, уточняется расположение биоэнергетических точек.

Противопоказаний у иглоукалывания очень мало: беременность, период обострения хронических заболеваний, признаки ОРВИ, инфекции и онкологические заболевания. В этих случаях воздействовать на организм нельзя.

Акупунктурная точка – это участок кожи и подкожной основы, который имеет комплекс взаимосвязанных микроструктур (кровеносных сосудов, нервов, клеток соединительной ткани и др.), создающих активную зону. Эта зона оказывает влияние на нервные окончания и осуществляет связь между участком кожи и внутренним органом. Область точки характеризуется усиленным поглощением кислорода, более высокой по сравнению с рядом лежащими участками кожи температурой, более низким электрическим сопротивлением. Диаметр активных точек изменяется в зависимости от состояния человека. У бодрствующего человека диаметр точек может составлять до 1 см, у спящего человека он меньше. В состоянии эмоционального напряжения и при острых заболеваниях площадь отдельных точек настолько увеличивается, что образуются большие участки кожи с повышенной электропроводностью.

Признаком правильного нахождения точки акупунктуры является появление при надавливании на область ее расположения особых ощущений, которые называются предусмотренными. Это те ощущения, которые ожидает получить врач при воздействии на точку. Пациент отмечает появление в области точки ощущения ломоты, тяжести, распирающего, болезненности, онемения, укола, о чем он и сообщает врачу. В зависимости от клинической картины заболевания врач ставит задачу добиться сильных, слабых

предусмотренных ощущений или ощущений средней силы. Иногда при воздействии на точку возникает иррадиация возникшего ощущения вверх, вниз, в сторону. Наглядное представление точек показано на рисунке 1.



Рисунок 1 – Представление точек акупунктуры

Наиболее часто при лечении методом рефлексотерапии используют точки, расположенные на линиях, которые древневосточные врачи называли каналами, или меридианами. Они обозначали их, используя названия органов (меридиан легких, сердца, желудка и т. д.). Также современные врачи часто пользуются цифро-цифровым или буквенно-цифровым обозначением точек, чтобы облегчить их запоминание. Меридиан обозначают римской цифрой, а также одной или двумя буквами английского, французского или немецкого алфавита. Арабской цифрой обозначен порядковый номер точки на меридиане. Массажистам и людям, не имеющим специального образования, но применяющим воздействие на точки акупунктуры с лечебной и профилактической целью, например методом точечного массажа, можно рекомендовать освоить цифро-цифровое обозначение точек. Использование цифро-цифрового обозначения точек при составлении методических пособий и атласов значительно облегчает процесс обучения врачей, впервые

осваивающих метод рефлексотерапии. Такое обозначение точек акупунктуры является целесообразным в методической литературе, рассчитанной на широкий круг читателей, и способствует распространению применения данного метода среди населения [1].

2.2 Исследование предметной области

Анализируя объем данных, разнообразие данных и методов можно отметить количественную сложность предметной области.

Всего существует 14 постоянных меридианов, где 12 парных меридианов делятся на ян-меридианы и инь-меридианы. Инь и ян выражают противоположные стороны явления, они находятся в состоянии противоборства, ограничивают и стараются вытеснить друг друга. Инь и ян никогда не находятся в покое, они постоянно дополняют и изменяют друг друга. По восточным представлениям, действует правило: «Если ян отступает, увеличивается инь, а если отступает инь, происходит увеличение ян». С позиции инь–ян можно рассматривать деятельность и покой, возбуждение и торможение, жар и холод, избыток энергии в меридиане и недостаток ее, избыточность функции органа и недостаточность этой функции. Инь ведает внутренней средой живого организма, ян – его внешними проявлениями. Концепция инь–ян может быть использована в западной медицине как модель целостного подхода к человеческому организму и пониманию состояния здоровья и болезни.

Постоянные классические меридианы имеют собственные акупунктурные точки в количестве от 9 до 67. Общее количество точек на 14 постоянных классических меридианах – 361.

Эффект лечения в большой степени зависит от метода воздействия на точку акупунктуры. Выделяют следующие методы воздействия – метод возбуждения/прибавления (тонизирующий) и метод торможения/отнятие (успокаивающий, седативный). При тормозном методе в меридиане избыточная энергия убавляется, при возбуждающем методе энергия добавляется.

Составлены общие положения, которым рекомендуют следовать при использовании этих методов. При тормозном методе рекомендуют вводить иглу медленно, кончик иглы направляют против течения энергии в меридиане, т.е. против хода меридиана. Воздействуют на относительно небольшое число точек. Время воздействия от 20-40 минут до нескольких часов. Извлекают иглу быстрым движением. При возбуждающем методе иглу вводят быстро, кончик иглы направляют по ходу меридиана. Вращение иглы осуществляют по ходу часовой стрелки. Воздействуют на достаточно большое количество точек. Время воздействия ограниченное от 30 секунд до 10 минут. Извлекают иглу медленно.

Предложено три вида раздражения точки с учетом глубины введения иглы времени воздействия: слабое, среднее и сильное. Среднему раздражению, которое называют еще гармонизирующим, врачи нередко отдают предпочтение, так как не всегда по клиническим симптомам можно определить, в каком состоянии находится меридиан.

При проведении лечения методами рефлексотерапии рекомендуют придерживаться общих принципов сочетания биологически активных точек, которые были разработаны еще древневосточными врачами.

Выбор биологически активных точек осуществляется, прежде всего, в соответствии с теорией меридианов. Практическое значение этой теории заключается в том, что лечение можно осуществлять там, где проходит меридиан, в котором выявлено нарушение, или воздействуя на точки смежного меридиана.

При лечении особое значение придают дистальным точкам. В дистальную группу точек входят стандартные точки меридианов.

Из проксимальных точек для воздействия на пораженный меридиан применяют сочувственные точки меридиана мочевого пузыря.

Считается, что точки, локализованные на голове, эффективны в основном при лечении заболеваний органов, расположенных в области головы

(заболевания глаз, ушей, носа, придаточных полостей носа); точки, локализованные в области груди, – при лечении заболеваний органов грудной полости, межреберной невралгии, мастита и мастопатии у женщин; точки, локализованные на спине, – при лечении остеохондроза позвоночника, артрита плечевого сустава, глубоко расположенных внутренних органов, связанных с тем или иным отделом позвоночника.

Нередко достаточно эффективным оказывается метод, основанный на использовании точек, находящихся в области пораженного органа. В то же время широко распространен метод воздействия на отдаленные точки. Например, отдаленные точки трех янских ручных меридианов эффективны при лечении головных болей и органов, расположенных в области головы. Отдаленные точки трех янских ножных меридианов используют при лечении патологических процессов, локализующихся в области головы и туловища. Если патологический процесс локализуется в области передней поверхности туловища, применяют точки меридиана желудка, в области боковой поверхности – точки меридиана желчного пузыря, в области спины – точки меридиана мочевого пузыря. Отдаленные точки трех иньских ручных меридианов рекомендуют при лечении заболеваний органов грудной клетки, трех иньских ножных меридианов – при лечении заболеваний органов брюшной полости и таза.

При выборе меридиан, который предполагается использовать для лечения, врач, как правило, рассматривают время активности выбранного меридиана. Если применяют седативный метод, то лечение будет более эффективным при воздействии на успокаивающую точку в период, когда активность энергии в меридиане достигает максимальной величины. Методом возбуждения рекомендуют воздействовать на тонизирующую точку меридиана в период его пассивности – в двухчасовой промежуток времени, который следует за периодом его наибольшей активности.

2.3 Обзор технических требований к интеллектуальной системе поддержки врача рефлексотерапевта

Интеллектуальная система – это совокупность аппаратного и программного обеспечения, которая способна имитировать или воспроизводить различные аспекты человеческого интеллекта и поведения для решения сложных задач, принятия решений и выполнения интеллектуальных действий.

Интеллектуальная система включает систему поддержки принятия решений: программные системы, которые помогают людям принимать решения на основе анализа данных.

Система имеет клиент-серверную структуру. Помимо звена системы управления базы данных и клиентского звена, выполняющегося в веб-браузере, имеется промежуточное звено – веб-сервер с соответствующим серверным программным обеспечением, представлена на рисунке 2.

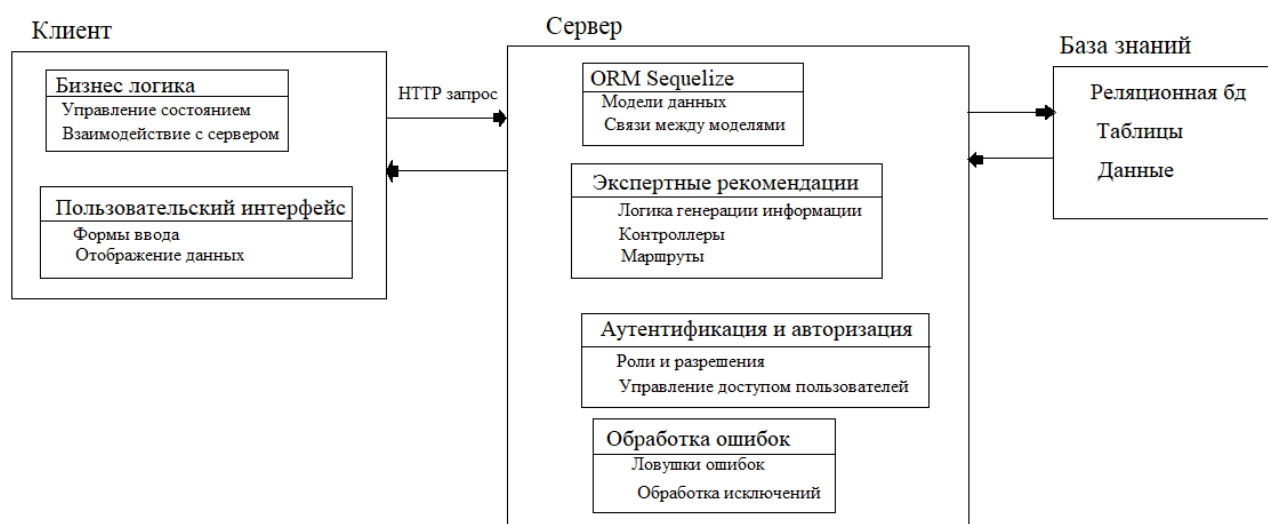


Рисунок 2 – Структура разрабатываемой системы

В качестве способа взаимодействия веб-приложения с сервером используем REST API.

API – это код, который позволяет двум приложениям обмениваться данными с сервера.

REST – это способ создания API с помощью протокола HTTP.

Технологию REST API применяют везде, где пользователю сайта или веб-приложения нужно предоставить данные с сервера. Например, после того как пользователь открыл приложение в браузере или нажал на какую-то кнопку, должно произойти какое-то действие, и мы должны получить данные с сервера. Мы отправляем по этому адресу запрос используя протокол HTTP. Сервер взаимодействует с базой данных, обрабатывает данные, после чего сервер возвращает данные на клиент, и пользователь получает данные в виде информации. Методы взаимодействия с сервером: GET(получить), POST(добавить), DELETE(удалить).

Одни из главных принципов REST API являются:

- отделение клиента от сервера. Клиент – это пользовательский интерфейс сайта или приложения. В REST API клиент содержит код, ответственный за обработку взаимодействия пользователей, создание запросов и отображение результатов, на сервере содержится код для доступа и управления базовыми данными, обеспечивая безопасность и эффективность. Это упрощает организацию API, позволяет легко переносить пользовательский интерфейс на другую платформу и дает возможность лучше масштабировать серверное хранение данных,

- единство интерфейса. Все данные должны запрашиваться через один URL-адрес стандартными протоколами, например, HTTP. Этот подход упрощает архитектуру сайта или приложения и делает взаимодействие с сервером понятнее.

Набор инструментов, которые будут использоваться на серверной части:

- Node JS в связке с Express. Express – это фреймворк для написания серверной части на Node JS. Node JS – это платформа на которой можно делать backend на языке JavaScript,

- в качестве системы управления базой данных будет использована PostgreSQL. Это объектно-реляционная СУБД. Преимущества, делающие систему управления базами данных универсальной и мощной, подходящей для

широкого спектра приложений, от небольших проектов до крупных корпоративных систем: открытый исходный код, что означает отсутствие лицензионных платежей; строгое соблюдение стандартов SQL, что обеспечивает совместимость и переносимость запросов между разными СУБД; система позволяет создавать собственные функции, операторы, типы данных и индексные методы, что дает гибкость в адаптации системы под специфические задачи; обеспечивает полную поддержку транзакций и соблюдает принципы ACID, что гарантирует надежность и целостность данных; система поддерживает широкий спектр встроенных типов данных, включая целые числа, числа с плавающей запятой, текстовые данные, даты и время, а также более сложные типы, такие как массивы; эффективно масштабируется, увеличивая мощности одного сервера и распределяя нагрузку между несколькими серверами, благодаря поддержке кластеризации и репликации данных,

- в качестве ORM для реляционных баз данных на Node JS используем Sequelize. ORM – это некая технология, которая позволяет связывать программный код с базой данных. Sequelize – это инструмент для организации взаимодействия между платформой Node JS и реляционными БД без использования специального языка SQL. Он связывает базы данных с объектами Java Script, создавая виртуальную объектную базу данных.

Набор инструментов, которые будут использоваться на клиентской части:

- React JS – это JavaScript-библиотека для создания пользовательских интерфейсов,
- React bootstrap – это библиотека React-компонентов, подходящих для повторного использования,
- для запроса к серверу используем библиотеку Axios. Это HTTP-клиент для Node JS и браузера,
- для навигации по страницам использоваться будет react-router-dom.

Блок экспертных рекомендаций отвечает за генерацию рекомендаций на основе данных, поступающих от пользователя и хранящихся в базе знаний. Он включает алгоритмы и логику, которые используют эти данные для формирования полезных рекомендаций.

Эта система использует машинное обучение для прогнозирования результатов лечения. Для создания модели «случайный лес» используется библиотека `ml-random-forest`.

Подсистема визуализации состоит из компонентов React. Компоненты - строительные блоки интерфейса. Например, «`symptomList`» и «`symptomDescription`», которые отображает список симптомов и позволяет пользователю искать и выбирать симптомы, и отображает подробную информацию о выбранном симптоме. Страницы являются компонентами высшего уровня, которые представляют собой разделы приложения. Например, «`symptom.js`» является страницей, где отображается список симптомов и их описание. Главный компонент приложения «`App.js`» объединяет все страницы и маршруты. Инициализация приложения осуществляется в «`index.js`».

База знаний – это организованный набор данных и информации, который содержит правила и закономерности, относящиеся к определенной области. Она используется для хранения и управления знаниями, а также для поддержки принятия решений, решения задач и автоматизации процессов. Базы знаний могут включать экспертные системы, базы данных, справочные материалы и другие ресурсы, которые помогают пользователям находить ответы на вопросы и решать проблемы в конкретных предметных областях. Можно сказать, что БЗ благоприятно влияет на повышение эффективности рабочего процесса, а именно: экономия времени при поиски нужной информации.

Нужно разработать базу знаний для интеллектуальной системы поддержки врача-рефлексотерапевта таким образом, чтобы были реализованы все необходимые функции для эффективной работы врача.

2.4 Обзор аналогов

Представлен обзор аналогов для выявления преимуществ и недостатков рассматриваемой информационной системы в области китайской медицины.

Информационная система «Shuboshi».

Данный аналог представляет из себя атлас акупунктурных точек китайской медицины и методы их использования для лечения популярных заболеваний.

Система включает:

- китайское название точек, представлено на рисунке 3,
- номер точки с обозначением меридиана на английском алфавите,
- графическое расположение точек,
- список заболеваний и их рецепты.

1. Болевая точка	1. Чи Цзе - P.5	1. Гань Шу - V.18	1. Чэн Фу - V.36
2. Ян Бай - VB.14	2. Цюй Цзе - MC.3	2. Дань Шу - V.19	2. Инь Мэнь - V.37
3. Цюань Ляо - IG.18	3. Сяо Хай - IG.8	3. Пи Шу - V.20	3. Вэй Чжун - V.40
4. Ди Цан - E.4	4. Нэй Гуань - MC.6	4. Вэй Шу - V.21	4. Чэн Шань - V.57
5. Ци Чжун - PC.41	5. Ле Цюе - P.7	5. Шэнь Шу - V.23	5. Шэнь Май - V.62
6. Шан Вань - VC.13	6. Тун Ли - C.5	6. Яо Ян Гуань - VG.3	6. Цзя Чэ - E.6
7. Чжун Вань - VC.12	7. Тай Юань - P.9	7. Да Чан Шу - V.25	7. Тай Юань - P.9
8. Ся Вань - VC.10	8. Да Лин - MC.7	8. Пан Гуан Шу - V.28	8. Ян Лин Цюань - VB.34
9. Чжан Мэнь - F.13	9. Шэнь Мэнь - C.7	9. Шан Ляо - V.31	9. Ян Цзяо - VB.35
10. Шуй Фэнь - VC.9	10. Юй Цзи - P.10	10. Да Чжуй - VG.14	10. Сюань Чжун - VB.39
11. Тянь Шу - E.25	11. Лао Гун - MC.8	11. Цзянь Вай Шу - IG.14	11. Кунь Лунь - V.60
12. Шэнь Цюе - VC.8	12. Сюе Хай - RP.10	12. Цзянь Цзин - VB.21	12. Цю Сюй - VB.40
13. Инь Цзяо - VC.7	13. Цюй Цюань - F.8	13. Фу Фэнь - V.41	13. Фэн Чи - VB.20
14. Ци Хай - VC.6	14. Инь Гу - R.10	14. Шэнь Чжу - VG.12	14. Тай Ян - BT
15. Шуй Дао - E.28	15. Инь Лин Цюань - RP.9	15. Чжи Ян - VG.9	15. Цянь Чжэн Сюе - PN.25
16. Гуань Юань - VC.4	16. Ди Цзи - RP.8	16. Цзи Чжун - VG.6	16. Тянь Цзун - IG.11
17. Гуй Лай - E.29	17. Сань Инь Цзяо - RP.6 (+)	17. Мин Мэнь - VG.4	17. Лян Цю - E.34
18. Чжун Цзи - VC.3	18. Тай Си - R.3	18. Шоу Сань Ли - GI.10	18. Сы Бай - E.2
19. Инь Лянь - F.11	19. Чжао Хай - R.6	19. Шао Хай - C.3	19. Нао Шан - PN.73
20. Цзу Сан Ли - E.36 (+)	20. Юн Цюань - R.1	20. Цюй Чи - GI.11	20. Си Ян Гуань - VB.33
21. Фэн Лун - E.40	21. Хуэй Инь - VC.1	21. Вай Гуань - TR.5	
22. Цзе Си - E.41	22. Дин Чуань - BT	22. Ян Лао - IG.6	
23. Тай Чун - F.3	23. Фэн Мэнь - V.12	23. Ян Чи - TR.4	
24. Нэй Тин - E.44	24. Фэй Шу - V.13	24. Хэ Гу - GI.4 (+)	
25. Син Цзянь - F.2	25. Синь Шу - V.15	25. Хуань Тяо - VB.30	

Рисунок 3 – Список точек в информационной системе «Shuboshi»

Преимущества системы Shuboshi: описаны методы для воздействие на точки, есть рецепты для лечения заболеваний и есть графическое расположение точек относительно тела человека, представлено на рисунках 4 и 5.

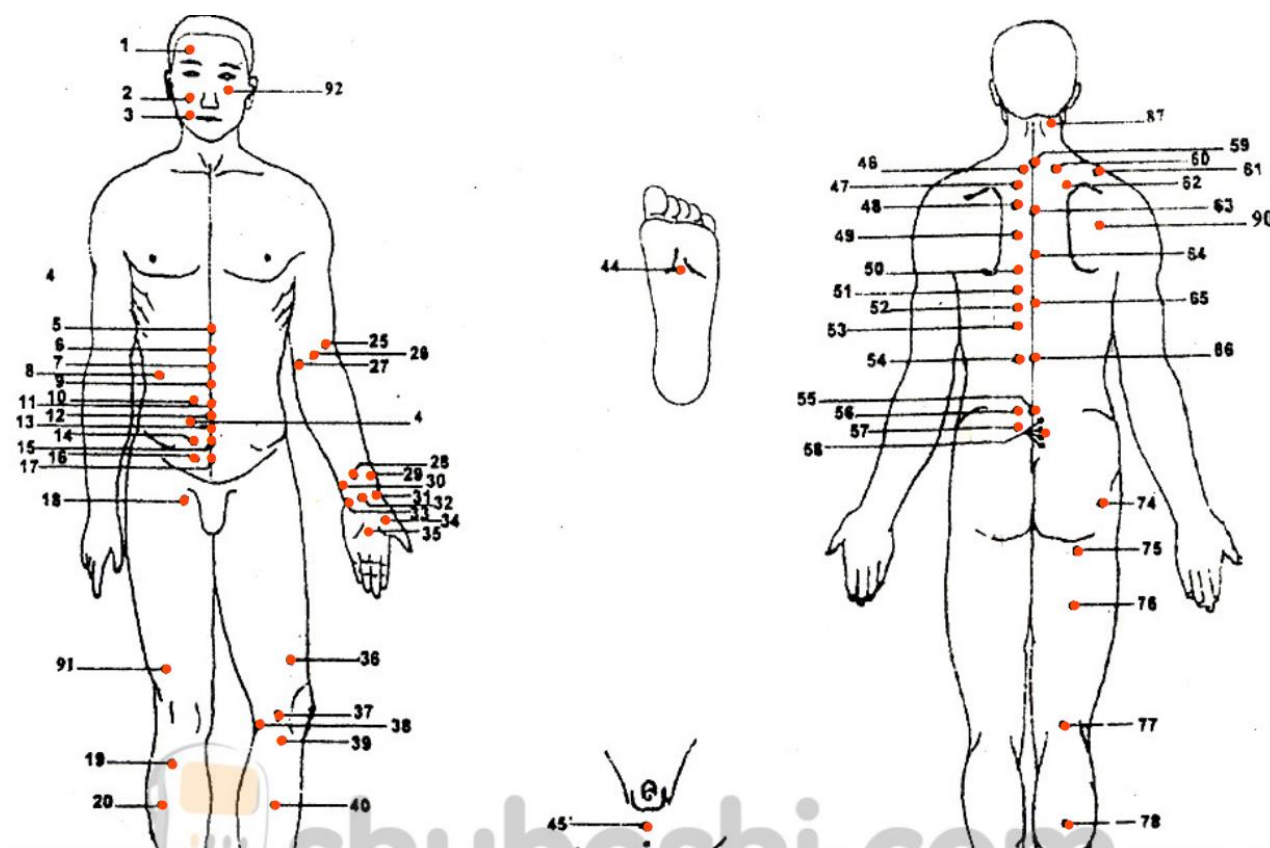


Рисунок 4 – Графическое отображение расположения точек на платформе «Shuboshi»

В качестве недостатков хочется подчеркнуть отсутствие поисковой строки, отсутствие информации о меридианах (позволяющая анализировать систему точек), отсутствие подробного описания метода воздействия, отсутствие описаний точек.

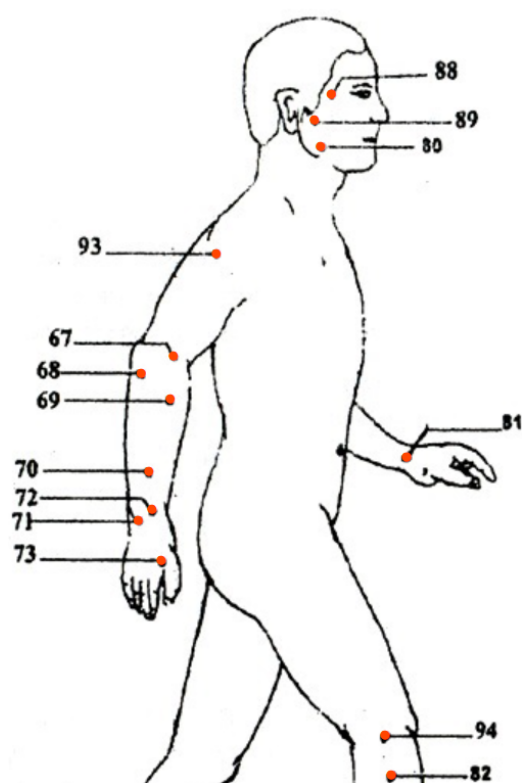
Приложение «Massage Maps».

Данный аналог представляет из себя рефлексологические схемы для массажа. В приложении показан эффективный альтернативный способ массажа с мануальным воздействием на области частей тела. Совокупность органов представлена картами, представлена на рисунке 6.

Приложение включает:

- схемы рефлексологии для разных частей тела,
- навигацию по картам,
- список органов и симптомов для каждой карты,

– возможность поиска нужного органа или симптома с переходом к соответствующей рефлекторной зоне на карте.



ГИПЕРТОНИЯ

Сочетания точек:

- Юн Цюань + Юн Цюань (44+44)
- Нэй Гуань + Цюй Чи (28+69) (с обеих сторон)
- Цзу Сан Ли + Сань Инь Цзяо (19+41)
- Сань Инь Цзяо + Юн Цюань (41+44)

Метод: каждую пару по 20 мин. на 1 или 2 режиме, 1 или 2 раза в день, сила воздействия умеренная.

СКЛЕРОЗ МОЗГОВЫХ АРТЕРИЙ

Сочетания точек:

- Фэн Чи + Цзянь Цзин (87+61)
- Сюань Чжун + Сань Инь Цзяо (84+41)
- Шэнь Шу + Шэнь Шу (54+54)

Метод: каждую пару по 10 мин. на 2,4-ом режиме, 1 или 2 раза в день, сила воздействия умеренная.

ОСЛОЖНЕНИЕ ПОСЛЕ ИНСУЛЬТА

Сочетания точек:

- Гань Шу + Гань Шу (50+50)
- Пи Шу + Пи Шу (52+52)

Рисунок 5 – Методы и рецепты лечения заболеваний на платформе «Shuboshi»

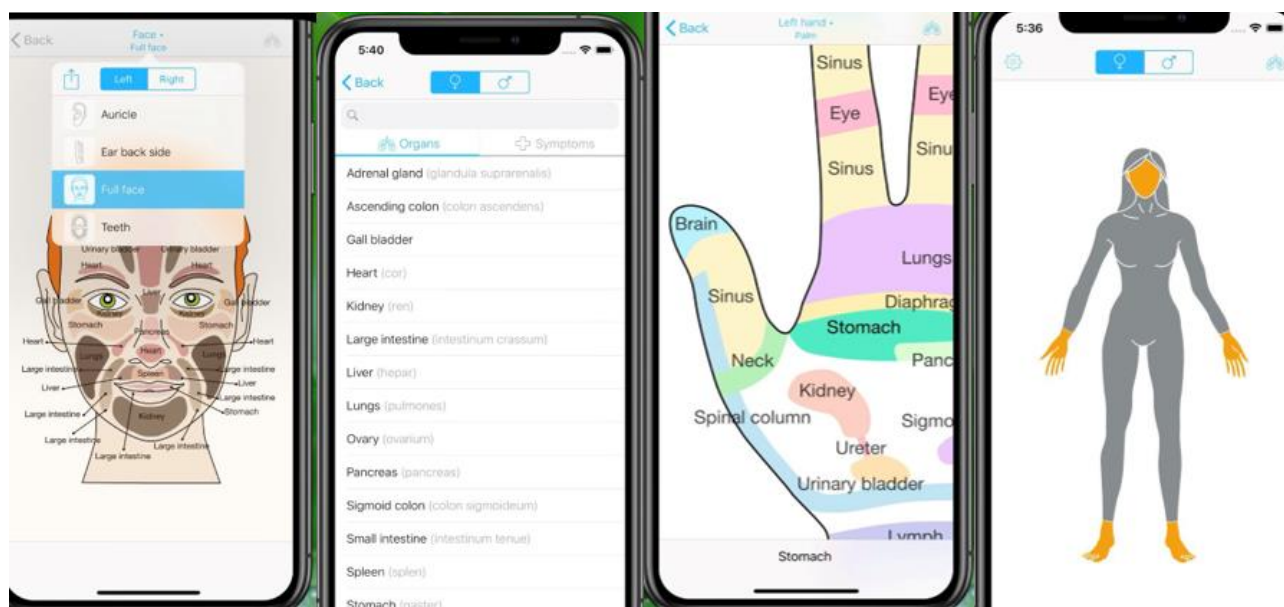


Рисунок 6 – Пример работы «Massage Maps»

В данном аналог доступно бесплатно лишь две области: стопы и кисти рук. Не несет никакой информации для обучения, только указания к массажу, без соответствующего метода. Иглоукальвание не рассматривается.

Плюс системы – выделение области воздействия цветом.

Приложение «Foot Reflexology Chart».

Это аналог представляет из себя набор областей точек стоп и с указанием принадлежности к ним симптомов. Есть список, по которому можно наглядно увидеть выделенные области точек при выборе симптома, представлен на рисунке 7.

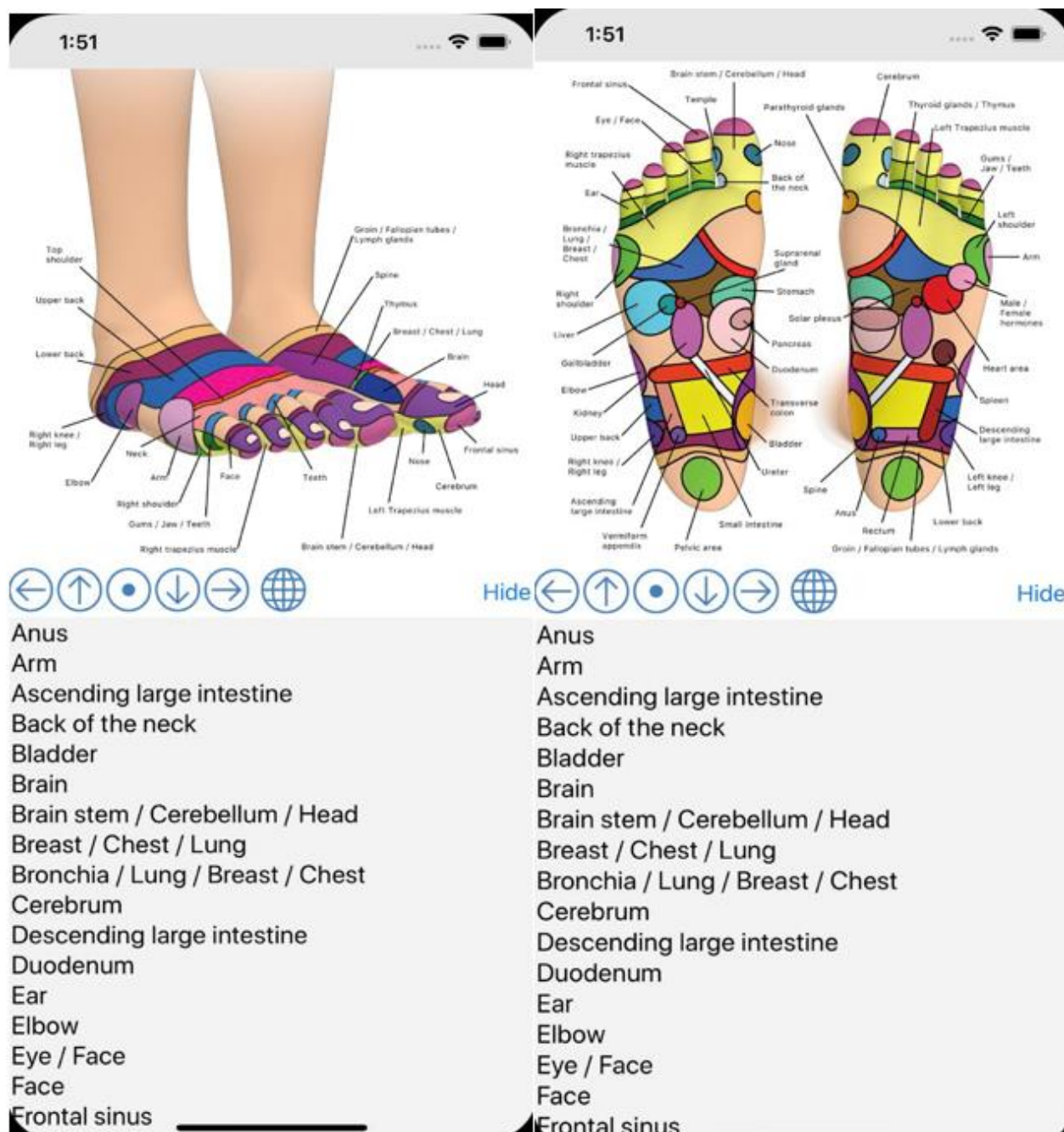


Рисунок 7 – Пример работы «Foot Reflexology Chart»

Преимущество этого аналога в представлении стоп в 3D модели. Но минусов намного больше, а именно: рассмотрение только стоп, нет поиска точек и симптомов по строке ввода, нет никаких пояснений и инструкций.

3 Разработка базы знаний

Конечной целью разработки является интеллектуальная система с единой подсистемой запросов к базе знаний, где в качестве значений будут выступать точки акупунктуры, методы, меридианы и заболевания, которые связаны между собой. Эта система будет использовать машинное обучение, чтобы предсказать результат лечения для последующих пациентов с аналогичными состояниями.

База знаний для интеллектуальной системы поддержки врача-рефлексотерапевта должна быть разработана таким образом, чтобы были реализованы все необходимые функции для эффективной работы врача. Учитывая существующие аналоги, нужно воплотить доступную и полную систему с простым и логически понятным интерфейсом.

Начинается разработка с формализации предметной области и выбора подходящей модели данных. Модель данных – это множество элементов (объектов, типов данных) и связей между ними, ограничений (например, целостности) и операций над объектами, типами данных и связями. Предлагается использовать реляционную модель для организации структурированных данных и сохранения связей между данными. В этой модели данные представлены в виде таблиц, состоящих из строк и столбцов. Отношение между таблицами устанавливается посредством ключей, которые связывают записи из разных таблиц. Реляционная модель позволяет добавлять новые таблицы и связи между ними без изменений существенной структуры. Модель из-за этого гибкая и легко расширяемая, что полезно при изменении и добавлении данных. Модель поддерживает целостность данных с помощью ограничений и связей, что обеспечивает связность данных и предотвращает возникновение противоречивых записей. И модель предоставляет операции для извлечения, изменения и анализа данных, что позволяет легко выполнить разнообразные запросы к базе данных, включая объединение, сортировку, фильтрацию и агрегации данных.

Для понимания основных сущностей предметной области и их взаимосвязей разработана инфологическая модель данных в виде диаграммы «Сущность-Связь» (ERD).

Инфологическая структура данных отражает концептуальную структуру данных и их взаимосвязи, представлена в виде сущностей, атрибутов и связей между ними, представлена на рисунке 8.

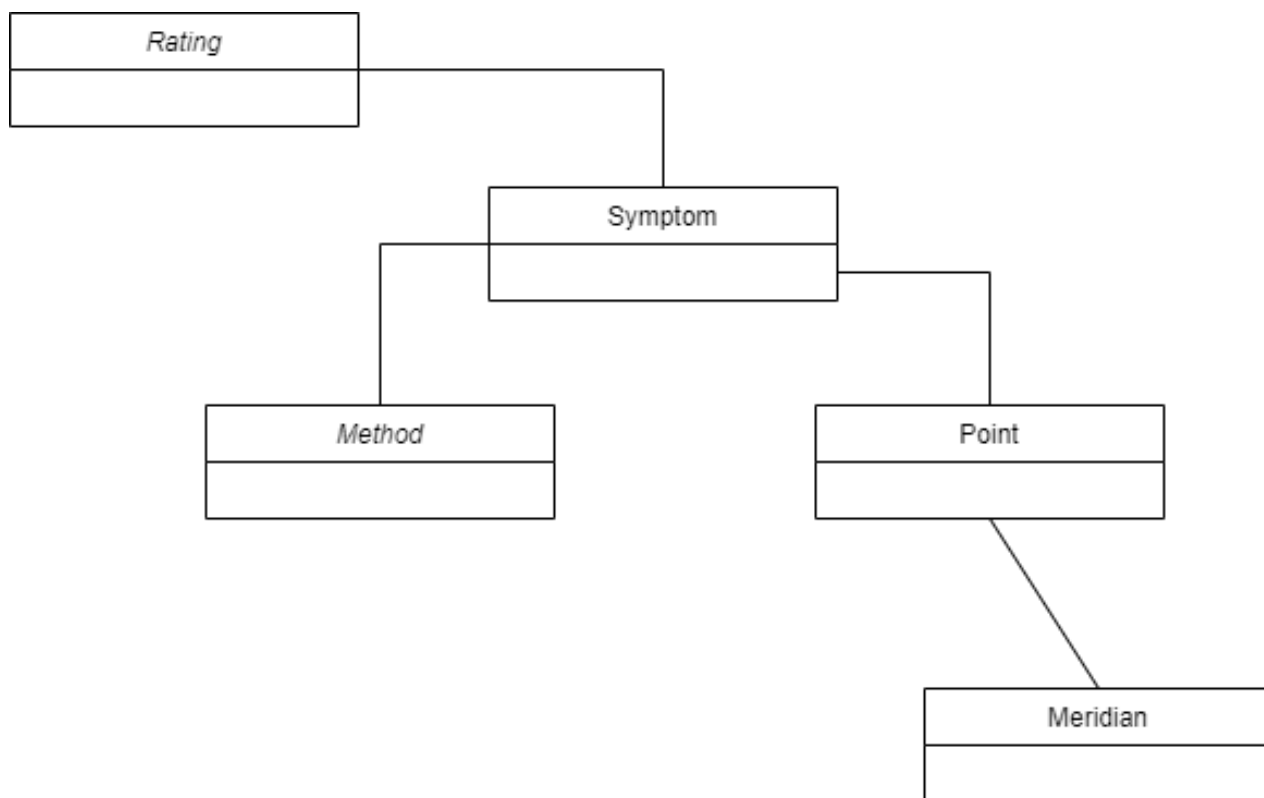


Рисунок 8 – Инфологическая модель структуры данных

При обращении к системе пользователь будет ориентироваться по 5 сущностям. «Симптом» позволяет решить проблемы пациента основываясь на его жалобах или диагнозе. «Рейтинг» хранит в себе оценки эффективности методов воздействия основанных на симптомах относительно возраста и пола пациента. «Точка» включает в себя список всех точек с подробными пояснениями. «Меридиан» представляет из себя совокупности точек по принципу китайской медицины. «Метод» – это способ, соответствующий лечению недомоганий с помощью точек. Каждую область можно рассмотреть, как познавательную.

Рассмотрим логическую структуру данных. Логическая модель представляет собой конкретизацию инфологической модели, адаптированную для реализации в СУБД, представлена на рисунке 9.

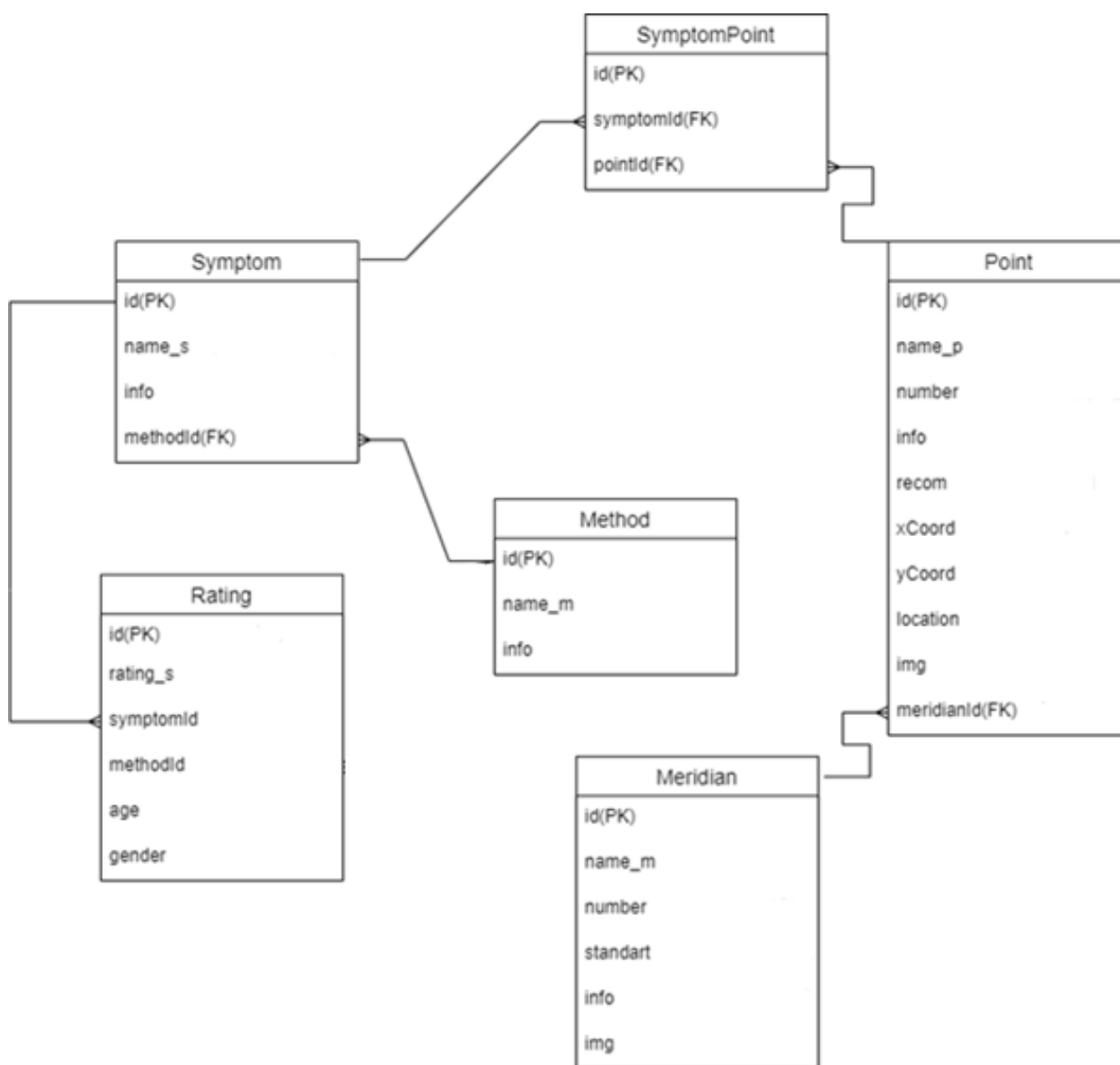


Рисунок 9 – Логическая модель структуры данных (PK- Primary key, FK – Foreign key)

Взаимосвязи данных: Meridian и Point имеют связь один ко многим, где один меридиан может иметь много точек, а точка принадлежит только одному меридиану; Method и Symptom также имеют связь один ко многим, один метод может быть связан с несколькими симптомами, а симптом относится только к одному методу; Rating и Symptom также имеют связь один ко многим, один симптом может быть связан с несколькими оценками, а оценка относится

только к одному симптому; Symptom и Point имеют связь многое ко многим, один симптом может быть связан с несколькими точками, одна точка может быть связана с несколькими симптомами. Взаимосвязь осуществляется через сущность SymptomPoint, которая содержит внешние ключи symptomId и pointId.

Сущность «Симптом» имеет название, информацию с дополнительным описанием недомогания. Сущность «Метод» имеет название и пояснительную информацию действий. Сущность «Точки» имеет название, номер, особенности и рекомендации точки если имеются, координаты расположения, описание расположения и изображение. Сущность «Меридиан» включает название, номер, информацию о меридиане, симптомы присущие меридиану и полную картинку со всеми точками входящими в меридиан. Сущность «Рейтинг» имеет оценку, возраст пациента и пол, и симптом по которому воздействовали.

Рассмотрим физическую структуру данных, включающая типы данных и их спецификацию, представлена на рисунке 10.

INTEGER – целочисленный тип данных. Id используются как идентификаторы, xCoord и yCoord используются как числовые значения. STRING – строковый тип данных. UNIQUE – параметр уникальности.

3.1 Разработка программного обеспечения для реализации запросов к базе знаний

Рассмотрим запросы разработанные для функционирования системы.

Связь запросов на сервере и клиенте осуществляется через REST API. Сервер предоставляет маршруты, которые клиент использует для получения данных. Рассмотрим пример о работе с данными о симптомах.

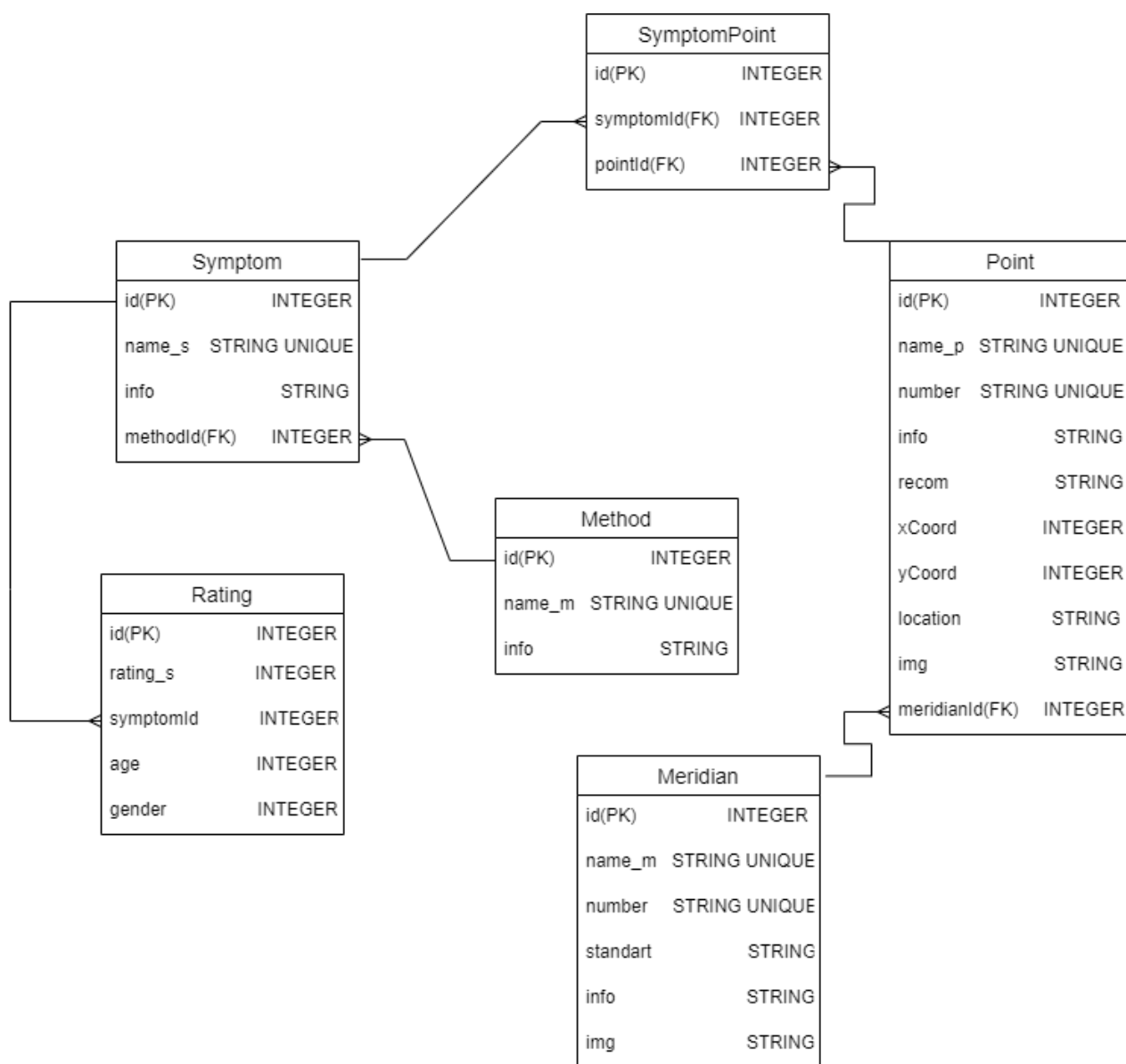


Рисунок 10 – Физическая модель структуры данных (PK- Primary key, FK – Foreign key)

Серверная часть.

Запросы к базе данных выполняются с использованием ORM Sequelize в Node.js.

В запросах используется метод «create». Метод принимает данные из тела запроса («req.body») и создает новую запись в таблице. Реализацию метода «create» можно увидеть на рисунке 11.

```

async create(req, res, next){
  try{
    const{name_s, info, pointss, methodId} = req.body
    const symptom = await Symptom.create({name_s,info, pointss, methodId})
    return res.json(symptom)
  } catch (e) {
    next(ApiError.badRequest(e.message))
  }
}

```

Рисунок 11 – Метод «create» в таблице Symptom

Следующий используемый метод «getAll» выбирает и возвращает все записи из таблицы с возможностью фильтрации по «methodId» и «id». Метод «getAll» продемонстрирован на рисунке 12.

```

async getAll(req, res){

  let {methodId, id, limit, page} = req.query
  page = page || 1
  limit = limit || 100
  let offset = page * limit - limit
  let symptoms;
  if (!methodId && !id){
    symptoms= await Symptom.findAndCountAll({limit, offset})
  }
  if(methodId && !id){
    symptoms = await Symptom.findAndCountAll( {where:{methodId}, limit, offset})
  }
  if(!methodId && id){
    symptoms = await Symptom.findAndCountAll( {where:{id}, limit, offset})
  }
  if(methodId && id){
    symptoms = await Symptom.findAndCountAll( {where:{methodId, id}, limit, offset})
  }
  return res.json(symptoms)
}

```

Рисунок 12 – Метод «getAll» в таблице Symptom

Метод «getOne» принимает «id» из параметров запроса («req.params») и возвращает запись из таблицы соответствующий «id». Метод «getOne» можно увидеть на рисунке 13.

```

async getOne(req,res){
  const {id} = req.params
  const symptom = await Symptom.findOne(
    {
      where: {id}
    },
  )
  return res.json(symptom)
}

```

Рисунок 13 – Метод «getOne» в таблице Symptom

Клиентская часть.

На клиенте используется библиотека axios для отправки http-запросов к серверу.

Запрос для получения данных о симптоме выполняется внутри компонента «SymptomDescription» с помощью функции «axios.get». Когда компонент получает идентификатор симптома «diseaseId», он отправляет первый запрос, чтобы получить основную информацию о симптоме. Реализация запроса для получения данных о симптоме показана на рисунке 14.

```

const response = await axios.get(`http://localhost:5000/api/symptom/${diseaseId}`);
setSymptomData(response.data);

```

Рисунок 14 – Запрос для получения данных о симптоме

Этот запрос выполняется к серверу по адресу «http://localhost:5000/api/symptom/\${diseaseId}». Запрос возвращает данные о конкретном симптоме в виде JSON-объекта, который сохраняется в состоянии «site» с помощью «setSymptomData(response.data)».

После получения данных о симптоме, компонент отправляет второй запрос для получения метода, связанного с этим симптомом, используя «methodId» из данных о симптоме. Реализация запроса для получения данных о методе, связанным с симптомом показана на рисунке 15.

```
const methodResponse = await axios.get(`http://localhost:5000/api/method/${response.data.methodId}`);
setMethodData(methodResponse.data);
```

Рисунок 15 – Запрос для получения данных о методе, связанным с симптомом

Этот запрос отправляется по адресу «http://localhost:5000/api/method/\${response.data.methodId}», где «response.data.methodId» – это идентификатор метода, связанного с ранее полученным симптомом. Запрос возвращает данные о методе, которые затем сохраняются в состоянии «methodData» с помощью «setMethodData(methodResponse.data)».

После этого отправляется третий запрос для получения списка точек, связанных с симптомом. Реализация запроса для получения данных о точках, связанных с симптомом показана на рисунке 16.

```
const spResponse = await axios.get(`http://localhost:5000/api/symptompoin?symptomId=${diseaseId}`);

const pointsData = await Promise.all(spResponse.data.rows.map(async (symptomPoint) => {
  const pointResponse = await axios.get(`http://localhost:5000/api/point/${symptomPoint.pointId}`);
  return pointResponse.data;
}));
```

Рисунок 16 – Запрос для получения данных о точках, связанных с симптомом

Этот запрос отправляется по адресу «http://localhost:5000/api/symptompoin?symptomId=\${diseaseId}». Затем для каждой точки выполняется дополнительный запрос по адресу «http://localhost:5000/api/point/\${symptomPoint.pointId}» для получения подробной информации о точке. Полученные данные о точках сохраняются в состоянии «spData» с помощью «setSPData(pointsData)».

Эти запросы выполняются последовательно внутри «useEffect», который отслеживает изменения в «diseaseId». И каждый раз, когда пользователь выбирает новый симптом, компонент автоматически обновляет свои данные, отправляя соответствующие запросы к серверу.

Опишем организацию запросов по подразделам. Обычно осуществляется путем создания отдельных маршрутов (routes) и контролеров (controllers) для каждого подраздела. Структура систем запросов показана на примере подраздела «symptom» на рисунке 17.

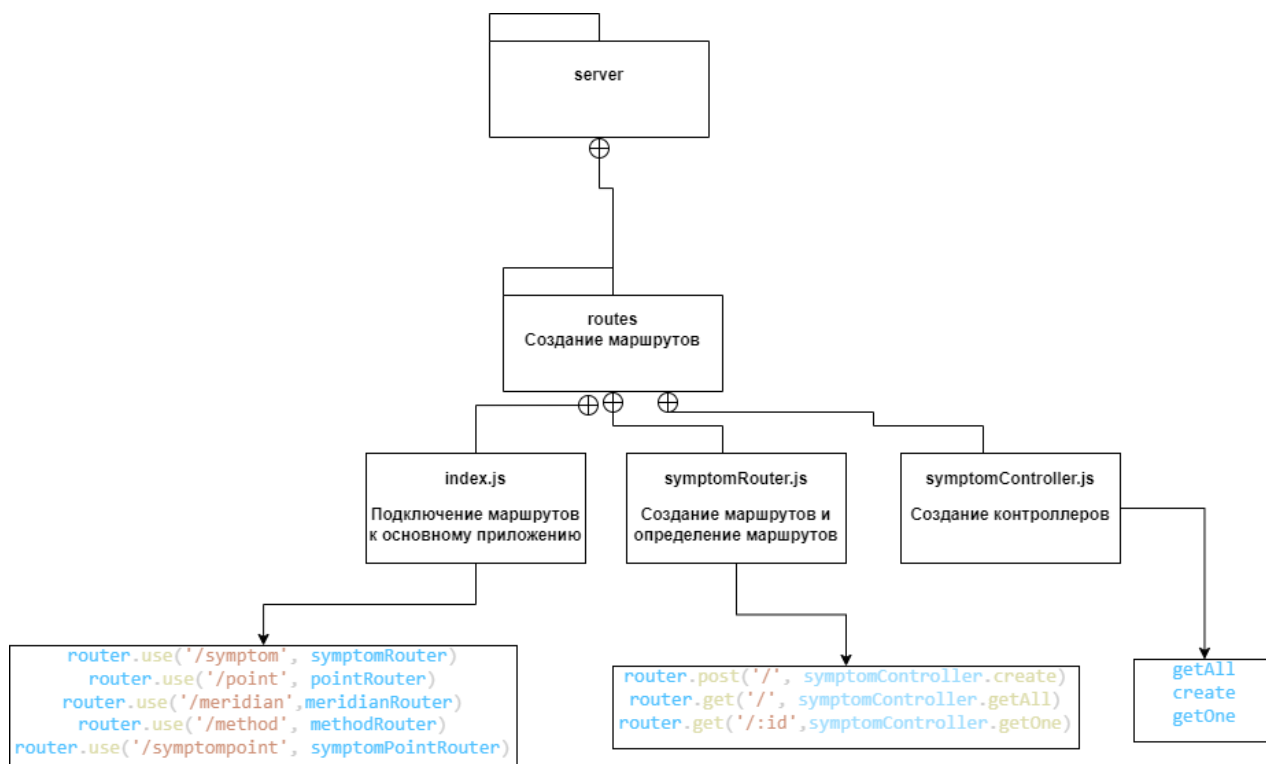


Рисунок 17 – Структура систем запросов подраздела «symptom»

Для каждого подраздела создаются отдельные файлы маршрутов.

В каждом файле маршрутов определяются соответствующие маршруты для выполнения операций (создание, чтение) связанные с конкретным подразделом.

Для каждого подраздела создаются соответствующие контроллеры, которые обрабатывают запрос к этим маршрутам. Контроллеры содержат логику обработки запросов и взаимодействия с базой данных или другими сервисами.

Запросы к подразделам организованы и обработаны отдельно, что делает код более структурированным.

3.2 Алгоритмы функционирования системы

Разрабатываемое программное обеспечение должно представлять собой web-приложение, которое включает в себя базу знаний, основанную на связанных данных для врача-рефлексотерапевта.

Программа должна обеспечивать выполнение следующих функций: отображение всех данных пользователю в удобном структурированном виде, принятие запроса пользователя к базе данных и отображение результата.

Входными данными являются запросы пользователя, которые поступают в базу данных: симптом (пользователь может найти способ излечения недомогания), точка (речь идет про акупунктурную точку, это составляющая способа излечения недомогания, где есть подробное описание ее расположения), метод (способ как именно нужно действовать при излечении), меридиан (совокупность точек по китайской медицине, которая поможет найти связи между ними и открыть новые исцеления).

Выходными данными являются результаты запроса, а именно описание решение проблемы пациента, или просто поясняющая информация. Данные будут поступать в структурированном виде, чтобы по содержанию было понятно описание.

Схема формирования рекомендаций для заданного симптома показана на рисунке 18.

При работе с сайтом пользователь получает обратную связь от своих действий. Это реализуется следующими способами: внешний вид элементов интерфейса изменяется при нажатии на них (иконки отображаются вдавленными или меняют оттенок на более темный); при отправке запроса выводится результат.

Схема сценариев взаимодействия с пользователем показана на рисунке 19.

Фильтрация данных. Пользователь вводит поисковый запрос, и данные фильтруются на основе этого запроса.

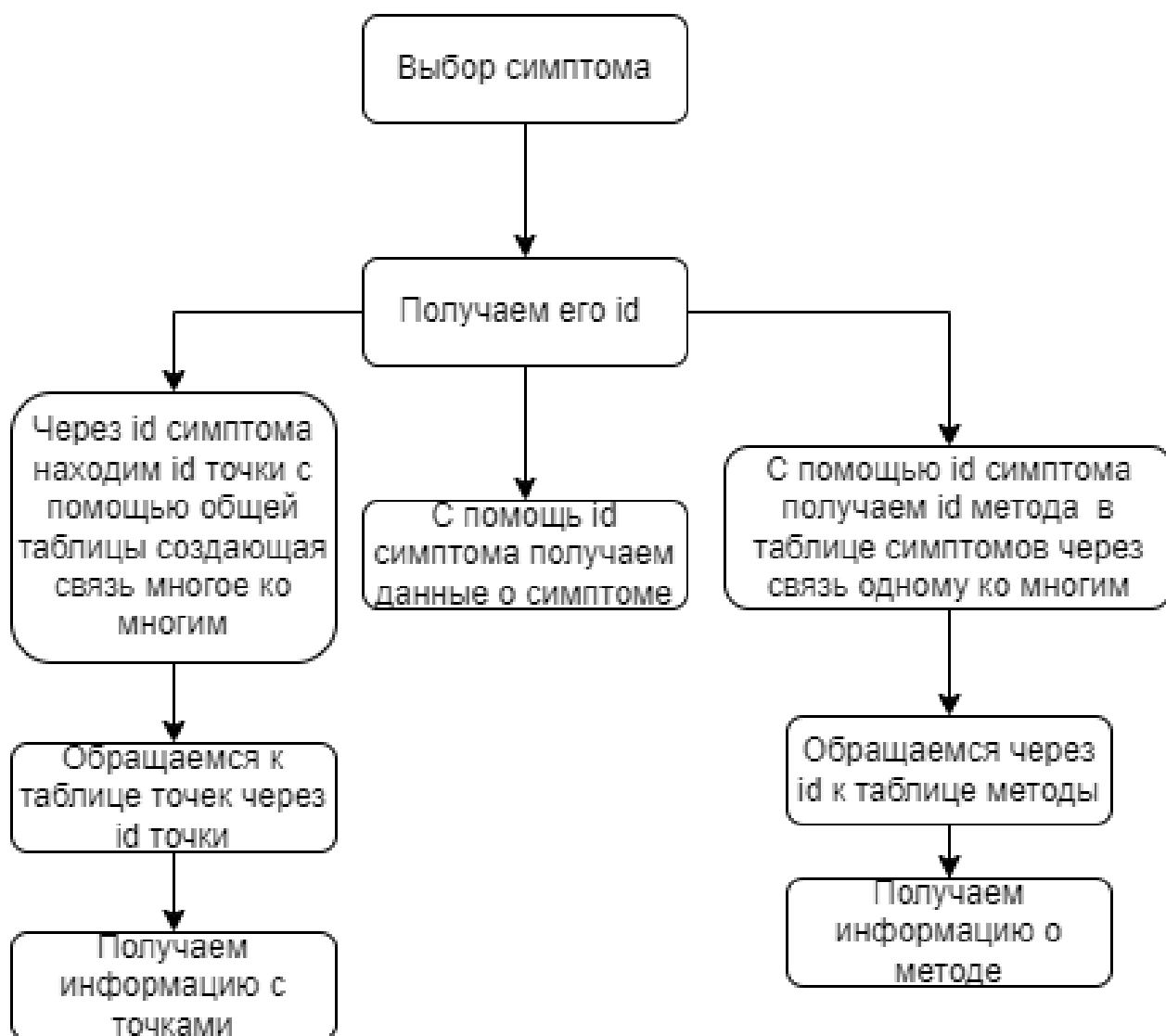


Рисунок 18 – Схема формирования рекомендаций для заданного симптома

Шаги фильтрации на примере:

- инициализация состояния поиска. Создается состояние «searchQuery» для хранения строки поиска, введенной пользователем,
- получение данных. Данные симптомов извлекаются из состояния глобального хранилища с использованием «Context»,
- обработка события изменения строки поиска. Когда пользователь вводит текст в поле поиска, это событие обновляет состояние «searchQuery»,
- фильтрация данных. Данные фильтруются на основе значения «searchQuery». Каждый элемент проверяется на соответствие строке поиска. Если строка поиска содержится в одном из указанных полей, элемент включается в отсортированный.

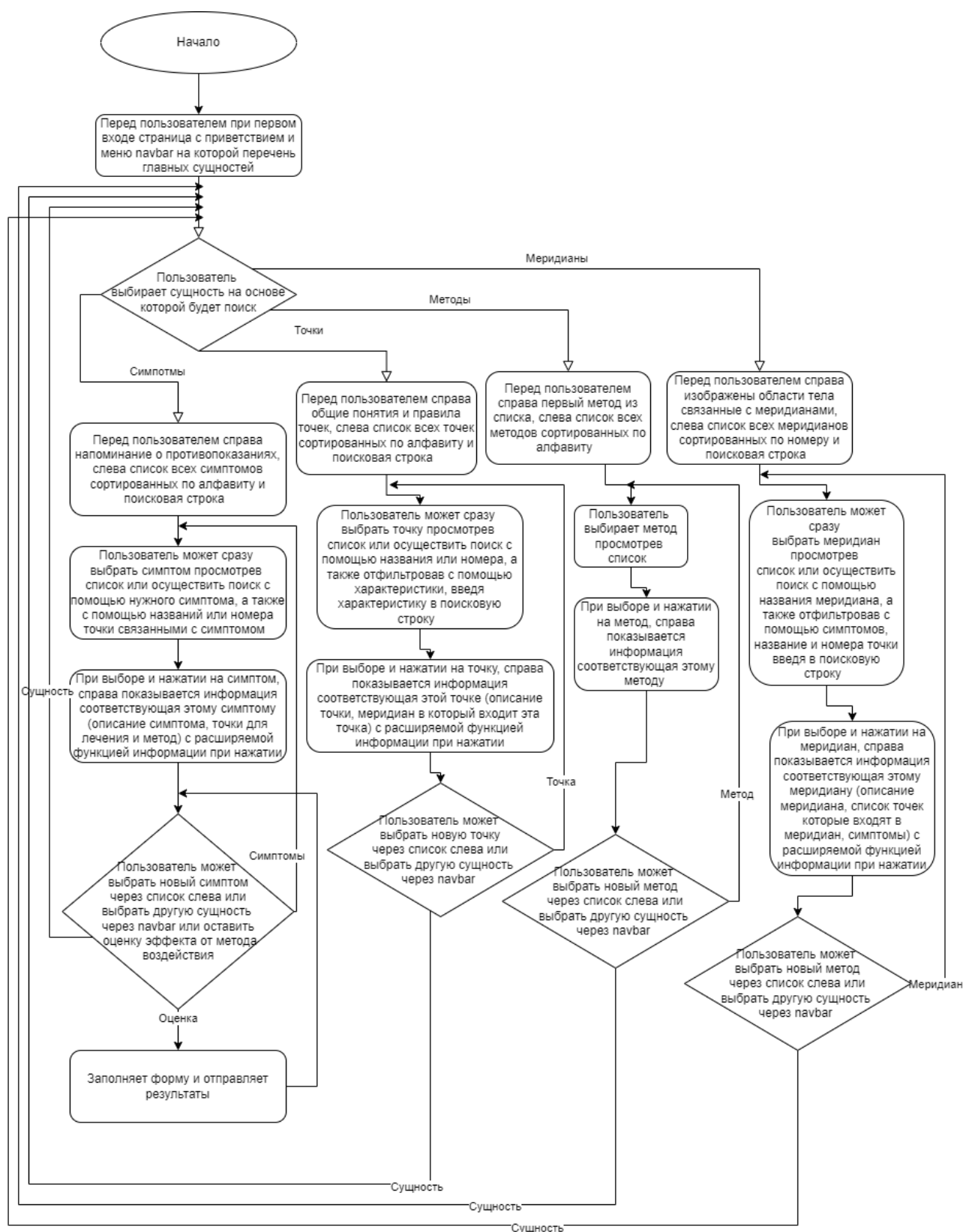


Рисунок 19 – Схема сценариев взаимодействия с пользователем

Для предсказания результата лечения для пациента при определенном симптоме в системе анализа используется алгоритм «случайный лес»,

являющийся методом машинного обучения. Схема функционирования модели показана на рисунке 20.

Цель построение такого алгоритма – это создание модели, которая предсказывает значения целевой переменной в зависимости о нескольких переменных на входе. Обучающими данными для модели являются данные пройденных приемов лечения.

В исходной выборке используются данные об объектах с известными значениями признака (symptom, age, gender) и целевой переменной (rating).

Для каждого дерева случайного леса используется подмножества обучающей выборки и случайное подмножество признаков.

Затем строятся деревья решений, разбивающие данные на основе признаков (age, gender, symptom) с целью минимизаций ошибки в оценке.

Прогнозы для каждого объекта получаются путем агрегации прогнозов от всех деревьев в лесу. В случае регрессии, прогнозы усредняются для получения окончательной оценки.

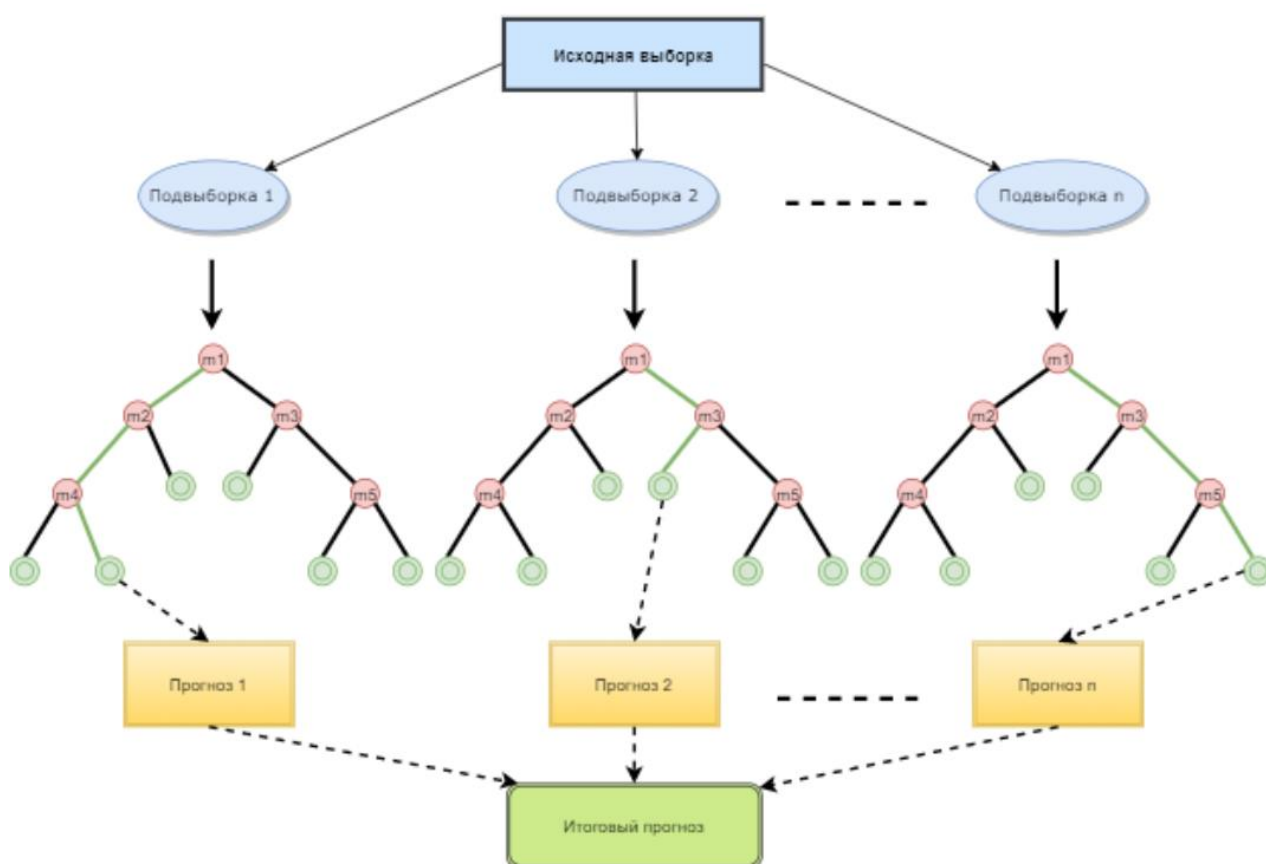


Рисунок 20 – Схема функционирования модели «случайный лес»

Итоговый прогноз возвращает окончательную оценку для каждого объекта, основанные на агрегированных прогнозах от всех деревьев в лесу.

Данные для анализа заполняются через форму на странице. Пользователь заполняет данные в форме на клиентской стороне (возраст, пол, симптом). Далее пользователь отправляет форму, данные из формы отправляются на сервер с помощью HTTP-запрос.

Сервер получает запрос с данными из формы. Сервер проверяет наличие необходимых данных и выполняется их валидация.

Если все данные валидны, сервер использует модель случайного леса для предсказания оценки. Модель «случайного леса» используется в классе «ForestModel», который продемонстрирован на рисунке 21.

Модель рассматривает такие факторы, как симптом, лечение, возраст. Модель учится на данных хранящихся в таблице «rating» в базе данных. Обучение модели показано на рисунке 22.

```
class ForestModel {
  constructor() {
    this.model = new RandomForestClassifier();
  }

  predict(symptomId, age, gender) {
    const prediction = this.model.predict([[symptomId, age, gender]]);
    return prediction;
  }

  train(features, labels) {
    this.model.train(features, labels);
  }
}
```

Рисунок 21 – Класс «ForestModel» и модель случайного леса

Когда предоставляется информация нового пациента, система использует функцию «predict» из класса «ForestModel» для получения прогнозируемого результата на основе обученной модели «случайный лес». Функция показана на рисунке 23.

```

class RatingController {
  async trainModel() {
    try {
      const ratings = await Rating.findAll();
      if (ratings.length === 0) {
        console.log('No ratings found. Skipping model training.');
        return;
      }
      const features = ratings.map(rating => [rating.symptomId, rating.age, rating.gender]);
      const labels = ratings.map(rating => rating.rating_s);

      this.forestModel.train(features, labels);
      console.log('Model trained successfully');
    } catch (e) {
      console.error('Error training model:', e);
    }
  }
}

```

Рисунок 22 – Обучение модели

Сервер отправляет результат предсказания обратно клиенту.

```

class RatingController {
  predict = async (req, res, next) => {
    try {
      //обучение модели при каждом запросе
      await this.trainModel()
      //извлечение данных из тела запроса
      const { symptomId, age, gender } = req.body;

      const symptom = await Symptom.findOne({ where: { id: symptomId } });

      if (!method || !symptom) {
        throw new Error('Method or Symptom not found');
      }

      //прогнозирование
      const prediction = this.forestModel.predict(symptomId, age, gender);
      return res.json({ prediction });
    } catch (e) {
      next(ApiError.internal(e.message));
    }
  }
}

```

Рисунок 23 – Получение и обработка данных на сервере, и прогнозирование на основе обученной модели

Схема алгоритма получения результата предсказания представлена на рисунке 24.

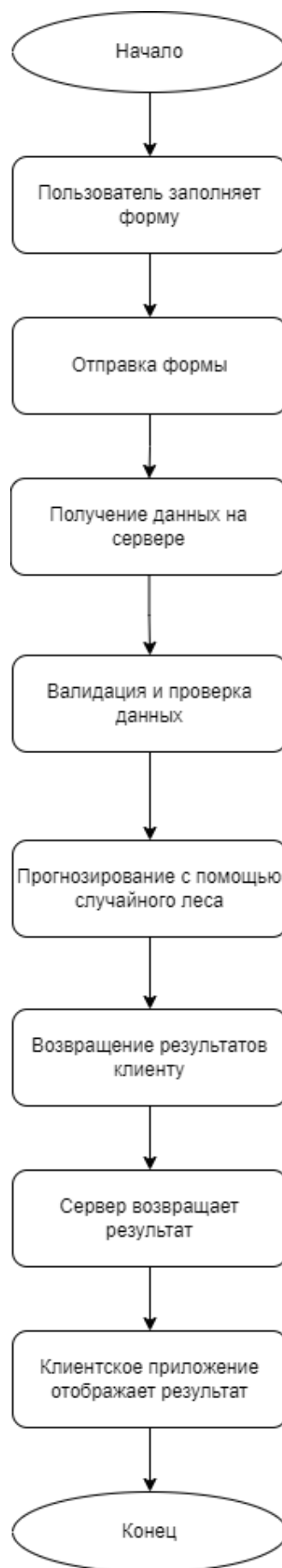


Рисунок 24 – Схема алгоритма получения результата предсказания

4 Разработка интерфейса пользователя и сценарии работы системы

Согласно проведённому анализу технического задания демонстрационное приложение должна иметь:

- графический интерфейс,
- структурированная подача данных,
- возможность принимать запросы от пользователя,
- отображение результата запроса.

Главная страница web-приложения будет выглядеть следующим образом, рисунок 25.

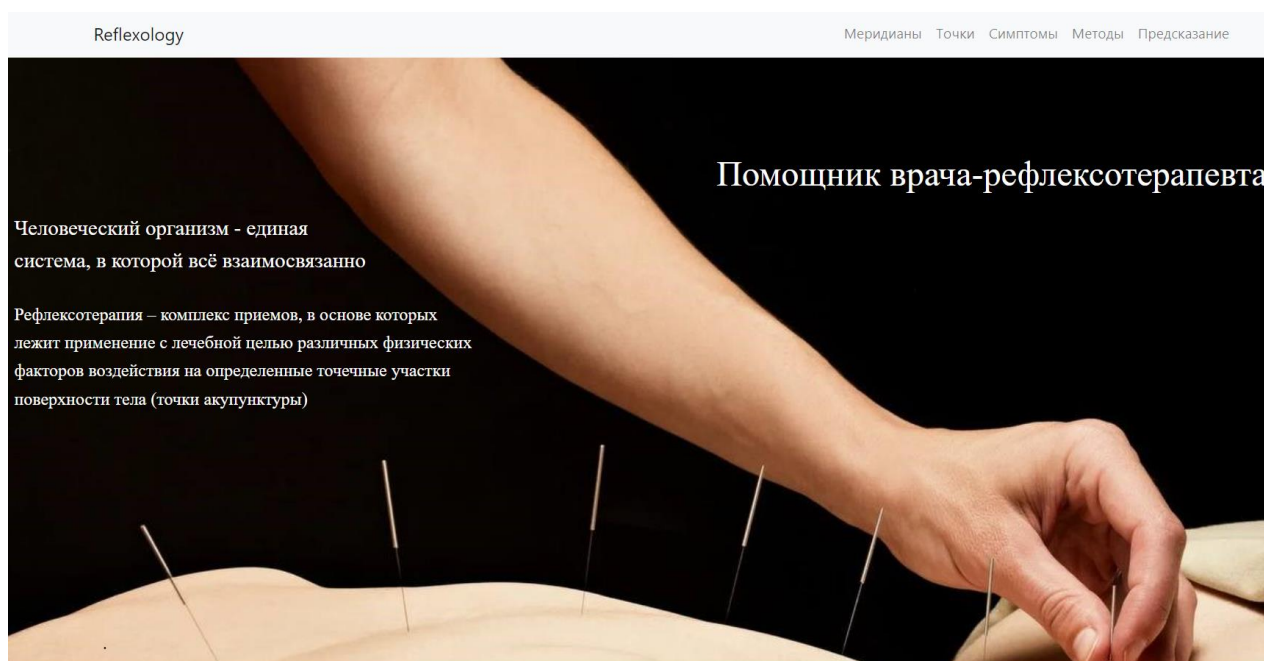


Рисунок 25 – Главная страница web-приложения

Перед пользователем приветственная страница, поясняющая что из себя представляет медицина, по которой сделано приложение. Сверху расположено меню паубар, где на выбор даны 5 сущностей.

Если выбрать сущность «Меридианы» пользователю будет предоставлен список всех меридианом и общая дополнительная информация (обозначение меридианов и взаимоотношение болевых ощущений и меридианов). Страница перехода на «Меридианы» будет выглядеть следующим образом, рисунок 26.

Поиск меридианов

Меридиан Легких

Меридиан Толстой
кишки

Меридиан Желудка

Меридиан
Селезенки-
Поджелудочной
железы

Меридиан Сердца

Меридиан Тонкой
кишкиМеридиан Мочевового
пузыря

Меридиан Почек

Меридиан Перикарда

Меридиан Трёх
обогревателейМеридиан Желчного
пузыря

Наиболее часто при лечении методом рефлексотерапии используют точки, расположенные на линиях, которые древневосточные врачи называли меридианами. Они обозначали их, используя названия органов (меридиан легких, сердца, желудка и т. д.). Также меридиан обозначают римской цифрой или одной или двумя буквами английского, французского или немецкого алфавита.

Взаимоотношение болевых ощущений и меридианов:

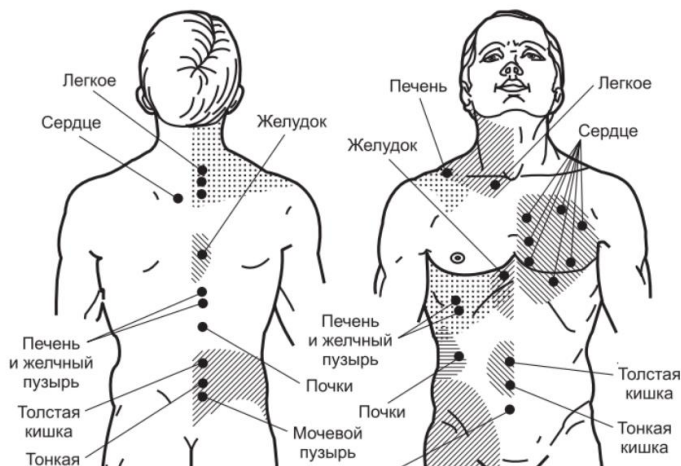


Рисунок 26 – Страница «Меридианы» при переходе с navbar

Выбирая один из списка меридиан, отображается информация соответствующая этому меридиану и выбранное подсвечивается как указание где сейчас находится пользователь. Демонстрация выбранного меридиана представлена на рисунках 27 и 28.

Поиск меридианов

Меридиан Легких

Меридиан Толстой
кишки

Меридиан Желудка

Меридиан
Селезенки-
Поджелудочной
железы

Меридиан Сердца

Меридиан Тонкой
кишкиМеридиан Мочевового
пузыря

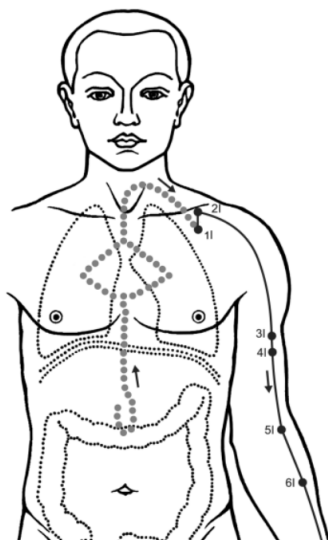
Меридиан Почек

Меридиан Перикарда

Меридиан Трёх
обогревателейМеридиан Желчного
пузыря

Меридиан Легких

(L, P, Lu)



Точки меридиана:

1I (чжун-фу)	Сигнальная
2I (юнь-мэнь)	
3I (тянь-фу)	
4I (ся-бай)	
5I (чи-цзе)	Седативная
6I (кун-цзуй)	Противоболевая
7I (ле-цзюе)	Стабилизирующий ло-пункт к меридиану толстой кишки
8I (цзин-цзюй)	
9I (тай-юань)	Тонизирующая, Пособник
10I (юй-цзи)	
11I (шао-шан)	

Рисунок 27 – Страница «Меридианы» при выборе меридиана

Reflexology

Меридиан Печени

Меридианы
Точки
Симптомы
Методы
Предсказание

Информация меридиана:

Меридиан легких – парный, относится к системе инь. Движение энергии в меридиане – центробежное, энергия идет от меридиана печени и переходит в меридиан толстой кишки. Период максимальной активности меридиана – от 3 до 5 часов, период минимальной активности – от 15 до 17 часов.

Меридиан начинается внутренним ходом в средней части туловища, затем поднимается вверх, проходит через диафрагму, легкие, гортань, далее спускается вниз в область плечевого сустава. Внутренний ход имеет ответвление, идущее от области легких к толстой кишке.

Наружный ход начинается от точки 11 чжун-фу (первое межреберье), поднимается к ключице, идет по передней поверхности плеча, по ладонно-лучевой линии предплечья, лучевому краю большого пальца, заканчивается у основания ногтя большого пальца точкой 111 шао-шан.

Симптомы меридиана:

кашель, одышка, ощущение удушья, другие симптомы заболевания легких и верхних дыхательных путей, бронхиальная астма, хронический тонзиллит боль в области грудной клетки, ключицы, спины, плеча, предплечья, в плечевом, локтевом и лучезапястном суставах; онемение в области передневнутренней поверхности руки; неврит лучевого нерва; воспалительные заболевания рта и глотки, кровоточивость десен, носовое кровотечение; неврит лицевого нерва; повышение температуры, ночные поты; кожные сыпи, кожные заболевания, сопровождающиеся зудом; головная боль, бессонница, тревожное состояние; заболевания сердца.

Рисунок 28 – Страница «Меридианы» при выборе меридиана(продолжение)

Пользователь может ознакомиться с описание меридиана, перечень симптомов на которые влияет меридиан, и есть список точек входящие в этот меридиан. На каждую точку можно нажать для отображения ее пояснений (описание расположения) и также при нажатии будет отображаться графически ее расположение на теле. Демонстрация графического изображения точке представлена на рисунке 29.

Reflexology

Меридианы
Точки
Симптомы
Методы
Предсказание

Поиск меридианов

Меридиан Легких

Меридиан Толстой кишки

Меридиан Желудка

Меридиан Селезенки-Поджелудочной железы

Меридиан Сердца

Меридиан Тонкой кишки

Меридиан Мочевого пузыря

Меридиан Почек

Меридиан Перикарда

Меридиан Трёх обогревателей

Меридиан Желчного пузыря

Меридиан Легких

(I, P, Lu)

Точки меридиана:

11 (чжун-фу)	Сигнальная
21 (юнь-мэнь)	
31 (тянь-фу)	
41 (ся-бай)	
51 (чи-цзе)	Седативная
61 (кун-цзуй)	Противоболевая
71 (ле-цюе)	Стабилизирующий ло-пункт к меридиану толстой кишки
81 (цзин-цюй)	
91 (тай-юань)	Тонизирующая, Пособник
101 (юй-цзи)	
111 (шао-шан)	

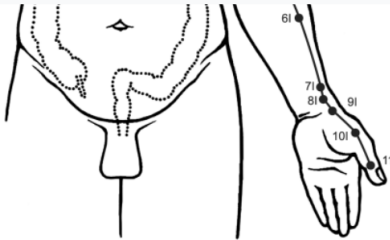
Рисунок 29 – Страница «Меридианы», графическое отображение точки меридиана при нажатии на нее из списка

Отображение информации точки на странице меридиана выглядит следующим образом, рисунок 30.

Reflexology

Меридиан Желчного пузыря

Меридиан Печени



Меридианы

Точки

Симптомы

Методы

Предсказание

111 (шао-шан)

Информация о выбранной точке:

Название: чжун-фу

Номер: 11

Расположение: В первом межреберье, на 6 цуней кнаружи от средней линии груди, на 2 цуня кнаружи от среднеключичной линии, ниже точки 21 юнь-мэнь на 3 см.

Сигнальная точка («точка тревоги», «точка-глашатай») позволяет определить функциональное состояние меридиана. Возникающая в этой точке боль является сигналом «тревоги» пораженного органа.

Информация меридиана:

Меридиан легких – парный, относится к системе инь. Движение энергии в меридиане – центробежное, энергия идет от меридиана печени и переходит в меридиан толстой кишки. Период максимальной активности меридиана – от 3 до 5 часов, период минимальной активности – от 15 до 17 часов.

Меридиан начинается внутренним ходом в средней части туловища, затем поднимается вверх, проходит через диафрагму, легкие, гортань, далее спускается вниз в область плечевого сустава. Внутренний ход имеет ответвление, идущее от области легких к толстой кишке.

Рисунок 30 – Страница «Меридианы», отображение информации точки меридиана при нажатии на нее из списка

При повторном нажатии на точку, она перестает выделяться и дополнительная соответствующая ей информация пропадает.

Поисковая строка позволяет фильтровать список. Фильтрацию списка можно осуществить через название меридиана. И также можно фильтровать с помощью симптомов, в списке остается только те меридиана у которых есть этот объект. Фильтрация списка меридианов по названию симптомов показана слева на рисунке 31, а фильтрация списка по симптомам показана слева на рисунке 32.

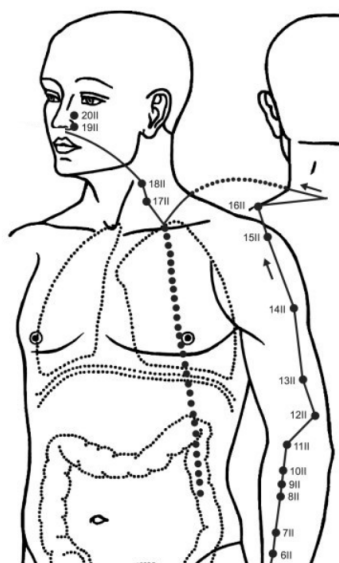
Также можно произвести фильтрацию за счёт точек входящих в меридиан через их китайские названия. Демонстрацию фильтрации списка меридианов по названию точек можно увидеть на рисунке 33.

Толстой кишки

Меридиан Толстой
кишки

Меридиан Толстой кишки

(II, GI, DI, LI)



Точки меридиана:

1II (шан-ян)

2II (эр-цзянь)

Седативная

3II (сань-цзянь)

4II (хэ-гу)

Пособник

5II (ян-си)

6II (пьянь-ли)

Стабилизирующий ло-пункт к меридиану легких

7II (вэнь-лю)

Противоболевая

8II (ся-лянь)

9II (шан-лянь)

10II (шоу-сань-ли)

11II (цзюй-чи)

Тонизирующая

12II (цзюнь-пан)

Рисунок 31 – Страница «Меридианы» фильтрация списка меридианов по названию меридиана

Кашель

Меридиан Легких

Меридиан Желчного
пузыря

Наиболее часто при лечении методом рефлексотерапии используют точки, расположенные на линиях, которые древневосточные врачи называли меридианами. Они обозначали их, используя названия органов (меридиан легких, сердца, желудка и т. д.). Также меридиан обозначают римской цифрой или одной или двумя буквами английского, французского или немецкого алфавита.

Взаимоотношение болевых ощущений и меридианов:

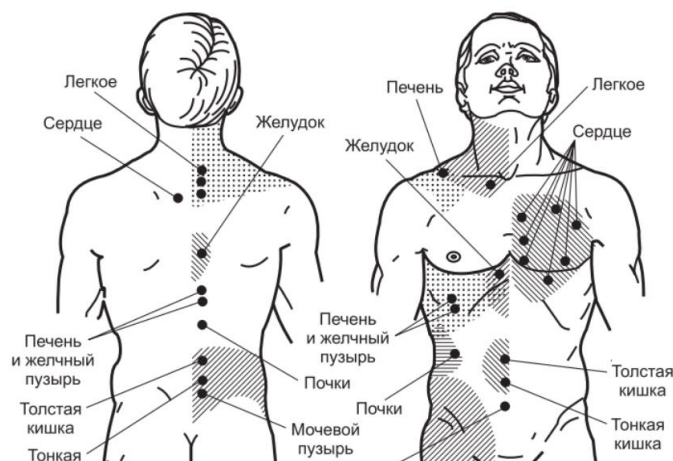


Рисунок 32 – Страница «Меридианы» фильтрация списка меридианов по симптомам

После можно перейти сущность «Точки». Раздел подразумевает полный список всех точек. Страница без выбранной точки рассказывает о основах

определения расположения и понятия точки. Страница перехода на «Точки» будет выглядеть следующим образом, рисунок 34.

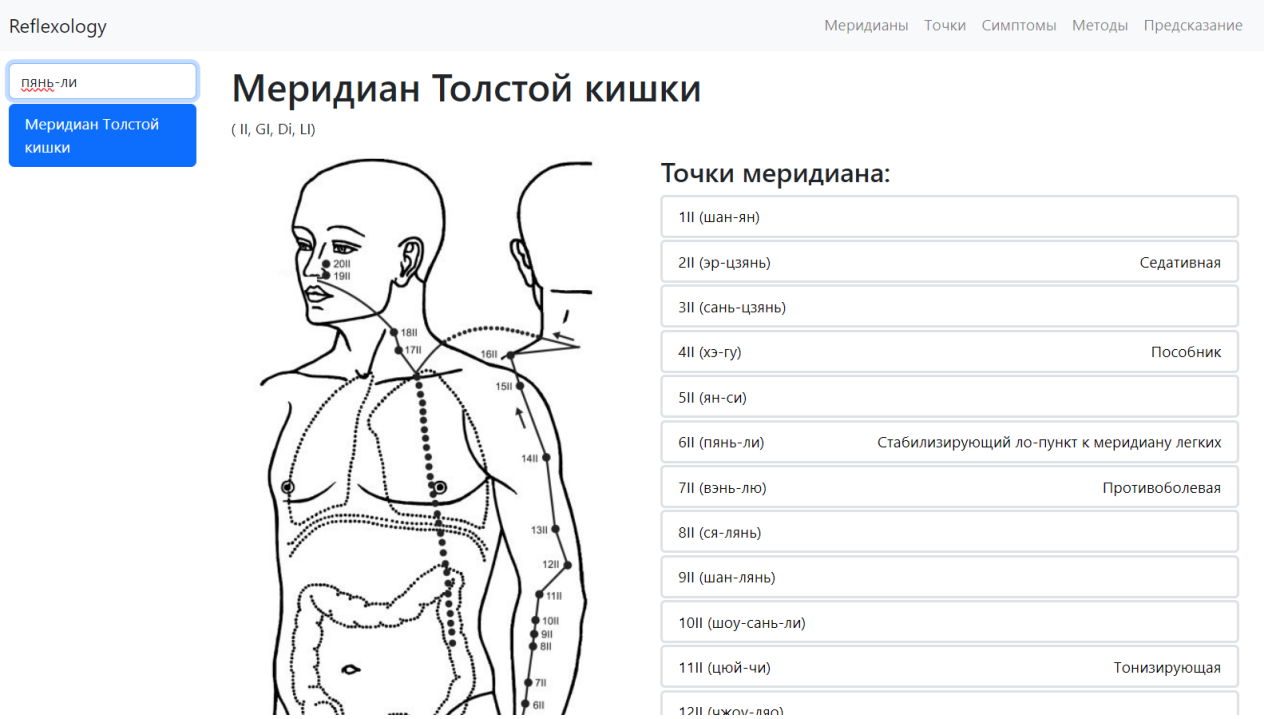


Рисунок 33 – Страница «Меридианы» фильтрация списка меридианов по названиям точек

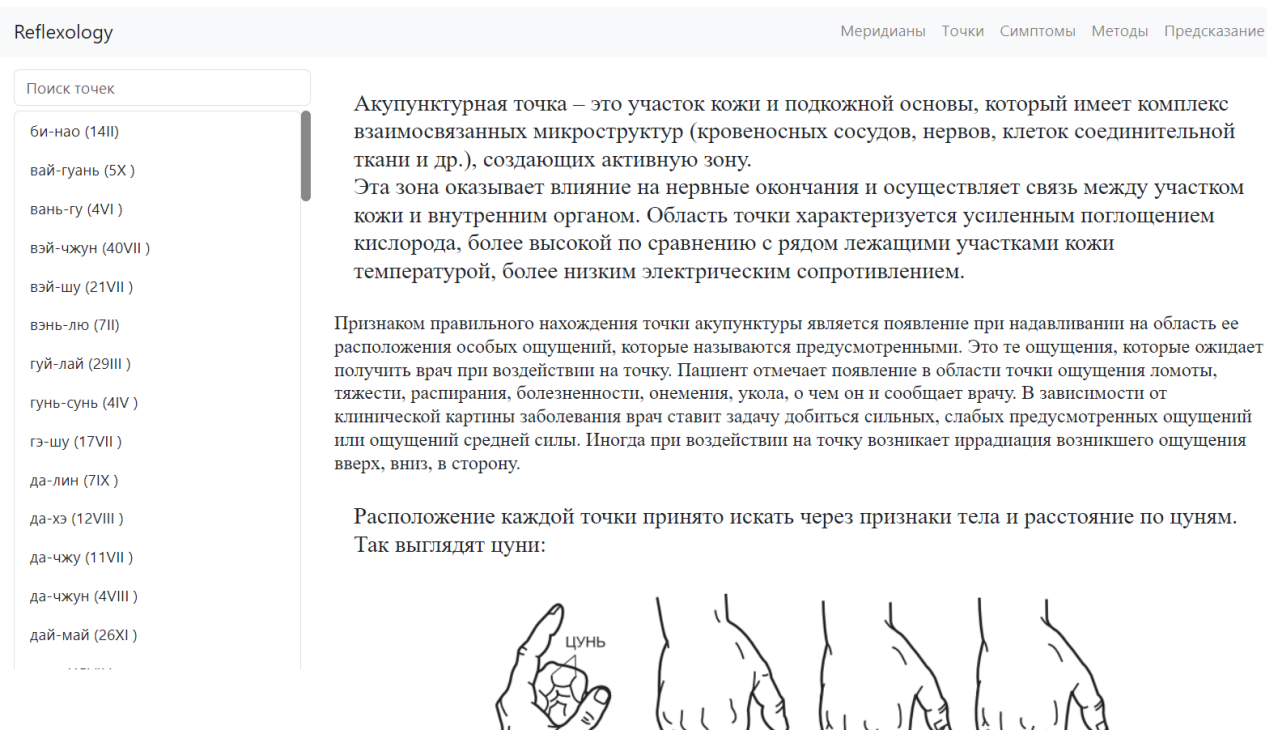


Рисунок 34 – Страница «Точки» при переходе с navbar

Выбирая одну из списка точку, отображается информация, соответствующая этой точке и выбранное подсвечивается как указание где

сейчас находится пользователь. Демонстрация выбранной точки представлена на рисунке 35.

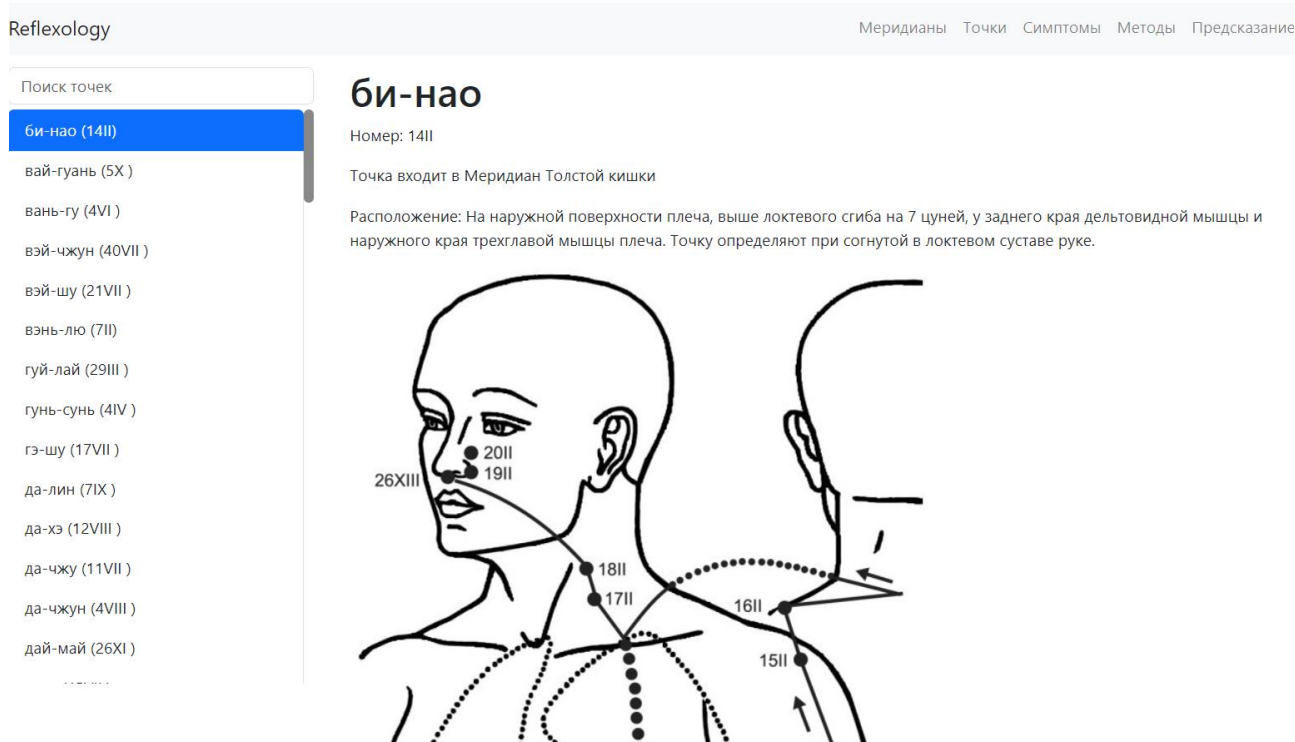


Рисунок 35 – Страница «Точки» при выборе точки

При наведении на информацию о меридиане, строка подчеркивается и появляется перо, указывающее что на объект можно нажать. После нажатия появляется информация о меридиане, которому принадлежит выбранная точка. Вкладку также можно закрыть нажатием на неё. Нажатие на меридиан, в который входит точка показано на рисунке 36.

Поисковая строка позволяет фильтровать список. Например, при вводе названия точки, в списке остается только те точки, у которых соответствуют названию. Фильтрация списка точек по названию показана слева на рисунке 37.

Также можно фильтровать через название характеристики точки. Фильтрация списка по характеристике показана слева на рисунке 38.

Поиск точек

би-нао (14II)

вай-гуань (5X)

вань-гу (4VI)

вэй-чжун (40VII)

вэй-шу (21VII)

вань-лю (7II)

гуй-лай (29III)

гунь-сунь (4IV)

гэ-шу (17VII)

да-лин (7IX)

да-хэ (12VIII)

да-чжу (11VII)

да-чжун (4VIII)

дай-май (26XI)

би-нао

Номер: 14II

Точка входит в Меридиан Толстой кишки

Меридиан толстой кишки – парный, относится к системе ян. Движение энергии в меридиане –центростремительное, энергия идет от меридиана легких и переходит в меридиан желудка. Период максимальной активности меридиана – от 5 до 7 часов, период минимальной активности – от 17 до 19 часов. Меридиан начинается наружным ходом чуть выше ногтевого ложа указательного пальца с лучевой стороны, идет вдоль этого пальца, между первой и второй пястными костями, по лучевой стороне предплечья, далее – по наружной поверхности плеча, в области плечевого сустава и верхней части спины. По трапецевидной мышце достигает надключной ямки и меридиан идет на шею, затем пересекает нижнюю челюсть, огибает губы и заканчивается в точке 20II ин-сян у наружного крыла носа с противоположной стороны. Внутренний ход идет вниз к легким, диафрагме и толстой кишке. На меридиане толстой кишки расположены 20 точек.

Расположение: На наружной поверхности плеча, выше локтевого сгиба на 7 цуней, у заднего края дельтовидной мышцы и наружного края трехглавой мышцы плеча. Точку определяют при согнутой в локтевом суставе руке.



Рисунок 36 – Страница «Точки» при нажатии на меридиан точки

да-ч

да-чжу (11VII)

да-чжун (4VIII)

да-чжу

Номер: 11VII

Точка входит в Меридиан Мочевыводящего пузыря

Расположение:

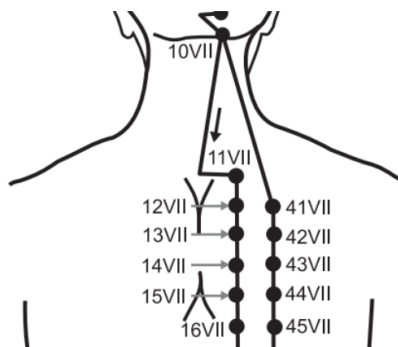


Рисунок 37 – Страница «Точки» фильтрация по названию точки

После можно перейти сущность «Симптомы». Раздел подразумевает полный список всех симптомов. Страница без выбранного симптома напоминает о противопоказаниях. Страница перехода на «Симптомы» будет выглядеть следующим образом, рисунок 39.

Сочувственная

вэй-шу (21VII)

сань-цзяо-шу (22VII)

шэнь-шу (23VII)

вэй-шу

Номер: 21VII

Точка входит в Меридиан Мочевого пузыря

Сочувственная точка для меридиана желудка

Расположение:

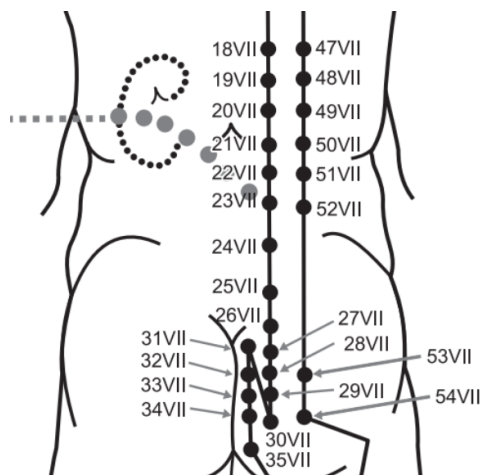


Рисунок 38 – Страница «Точки» фильтрация по названию характеристики точки

Поиск симптомов

Боль в голеностопном суставе
 Боль в запястье
 Боль в локтевом суставе
 Боль в тазобедренном суставе
 Вегето-сосудистая дистония
 Депрессия
 Заболевания артерий
 Заболевания вен
 Заболевания глаз
 Заболевания кожи
 Заболевания костей
 Заболевания мышц
 Заболевания носа
 Заболевания органов малого таза

В этих случаях взаимодействовать на точки нельзя, и иного варианта быть не может!

Противопоказания: беременность, период обострения хронических заболеваний, признаки ОРВИ, инфекции и онкологические заболевания.

Рисунок 39 – Страница «Симптомы» при переходе с navbar

При переходе на один из симптомов, в правой части экрана показывается информация для излечения этого недомогания. Демонстрация выбранной точки представлена на рисунке 40.

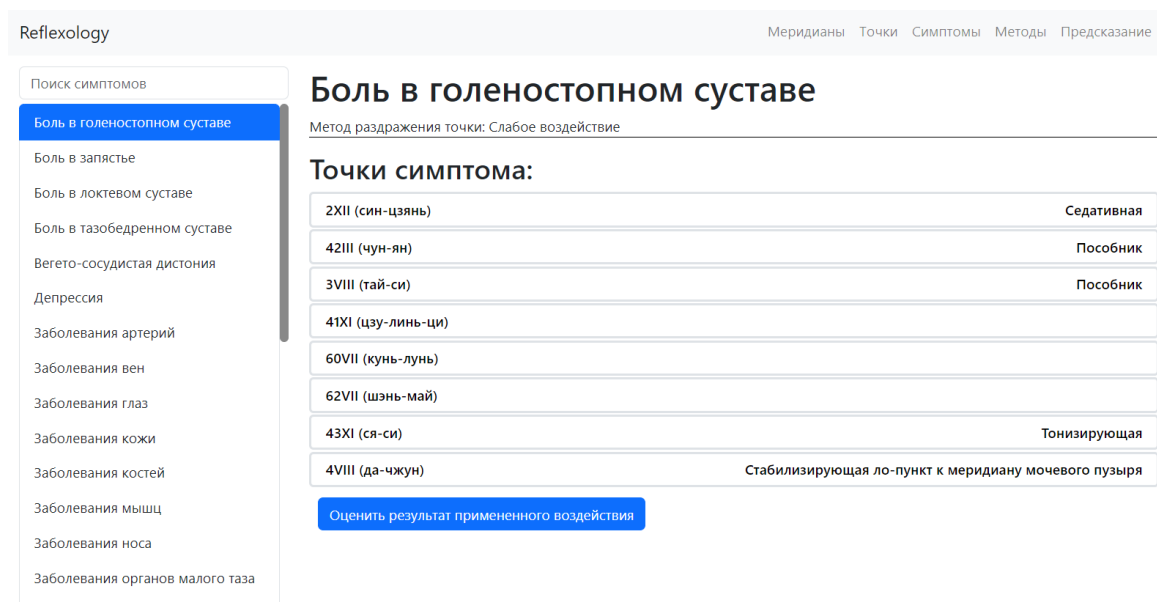


Рисунок 40– Страница «Симптомы» выбор одно из симптомов

При наведении на информацию о методе, строка подчеркивается и появляется перо, указывающее что на объект можно нажать. После нажатия появляется информация о методе, которая говорит, как именно лечить недомогание. Вкладку также можно закрыть нажатием на неё. Раскрытие метода соответствующий симптому показан на рисунке 41.

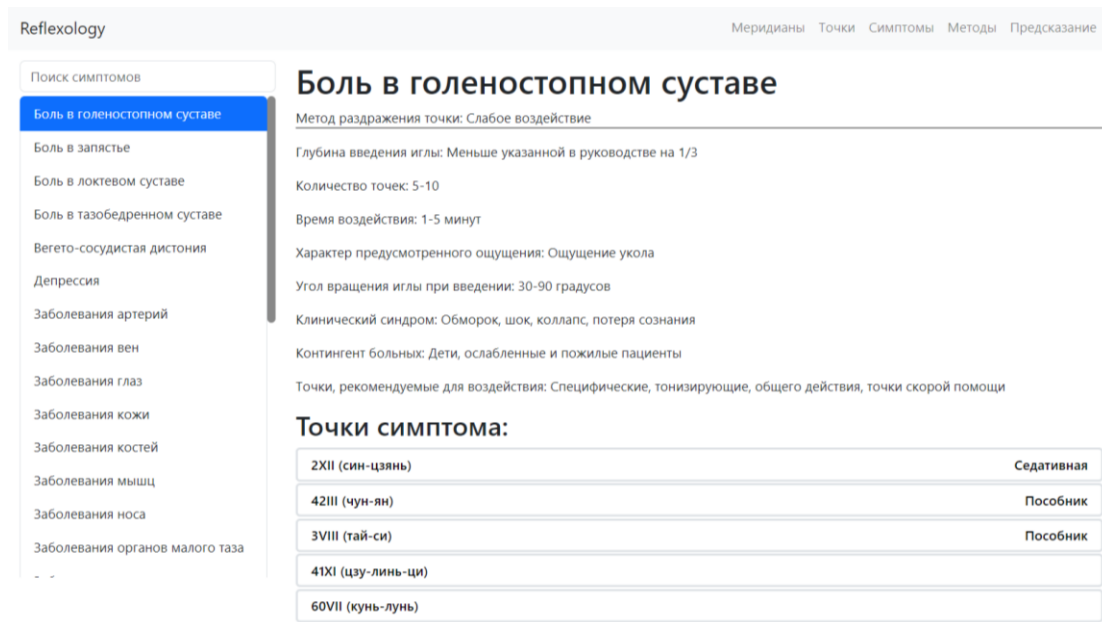


Рисунок 41– Страница «Симптомы» раскрытия метода соответствующий симптому

На каждую точку можно нажать для отображения ее пояснений (описание расположения) и также при нажатии будет отображаться графически ее

расположение на теле. При повторном нажатии информация убирается, и кнопка перестает выделяться. Раскрытие описания точки показано на рисунке 42.

Reflexology

Меридианы Точки Симптомы Методы Предсказание

Поиск симптомов

Боль в голеностопном суставе

Метод раздражения точки: Слабое воздействие

Точки симптома:

2XII (син-цзянь)	Седативная
42III (чун-ян)	Пособник
3VIII (тай-си)	Пособник
41XI (цзу-линь-ци)	
60VII (кунь-лунь)	
62VII (шэнь-май)	
43XI (ся-си)	Тонизирующая
4VIII (да-чжун)	Стабилизирующая ло-пункт к меридиану мочевого пузыря

Описание точки:

Номер: 2XII

Седативная (успокаивающая) точка оказывает угнетающее, успокаивающее действие на меридиан. Рекомендуется воздействовать на нее методом торможения.

Расположение: Во впадине между первым и вторым плюснефаланговым суставом.

Рисунок 42 – Страница «Симптомы» раскрытия точки соответствующий симптому

Поисковая строка позволяет фильтровать список. Например, при вводе названия симптома, в списке остается только те болезни, у которых соответствуют названию. Фильтрация списка симптомов по названию показана слева на рисунке 43.

Reflexology

Меридианы Точки Симптомы Методы Предсказание

Почечнокаменная болезнь

Почечнокаменная болезнь

Почечнокаменная болезнь

Метод раздражения точки: Слабое воздействие

Точки симптома:

6VIII (чжао-хай)	
10II (шоу-сань-ли)	
3VIII (тай-си)	Пособник
22VII (сань-цзяо-шу)	Сочувственная для меридиана трёх обогревателей
10VIII (инь-гу)	
24III (хуа-жоу-мэнь)	
28III (шуй-дао)	
23VII (шэнь-шу)	Сочувственная для меридиана почек
52VII (чжи-ши)	

Оценить результат примененного воздействия

Рисунок 43 – Страница «Симптомы» фильтрация по названиям симптомов

Или при вводе названия точки, в списке остается только те болезни, у которых соответствуют названия точки в рецепте. Фильтрация списка симптомов по названию точек показана слева на рисунке 44.

Reflexology

Меридианы Точки Симптомы Методы Предсказание

да-

Боль в голеностопном суставе

Боль в запястье

Боль в голеностопном суставе

Метод раздражения точки: Слабое воздействие

Точки симптома:

2XII (син-цзянь)	Седативная
42III (чун-ян)	Пособник
3VIII (тай-си)	Пособник
41XI (цзу-линь-ци)	
60VII (кунь-лунь)	
62VII (шэнь-май)	
43XI (ся-си)	Тонизирующая
4VIII (да-чжун)	Стабилизирующая ло-пункт к меридиану мочевого пузыря

Описание точки:

Номер: 4VIII
 Стабилизирующая точка (ло-пункт) обеспечивает равновесие в спаренных меридианах. От ло-пункта идет вторичный меридиан, осуществляющий соединение со спаренным меридианом.

Расположение: Кпереди от места прикрепления ахиллова сухожилия к пяточной кости, ниже точки 3VIII тай-си на 1,5 см

Рисунок 44 – Страница «Симптомы» фильтрация по названиям точек

Для каждого симптома можно оценить результат применяемого метода воздействия. Оценивается результат от отсутствия эффекта до максимально возможного результата лечения. Для каждого человека воздействие может сработать по-разному, поэтому дополнительные параметры — это пол и возраст человека. Для сохранения введенных данных нужно нажать «Сохранить», для отмены отправки достаточно закрыть форму. Накопленная информация будет использована для прогнозирования результата определенного симптома для человека. Форма для заполнения данными о результате уже применённого лечения показана на рисунке 45.

Далее можно перейти к сущности «Метод». Показывается поясняющая информация каждого метода. Сделан как информационно обучающий источник. Страница перехода на «Методы» будет выглядеть следующим образом, рисунок 46.

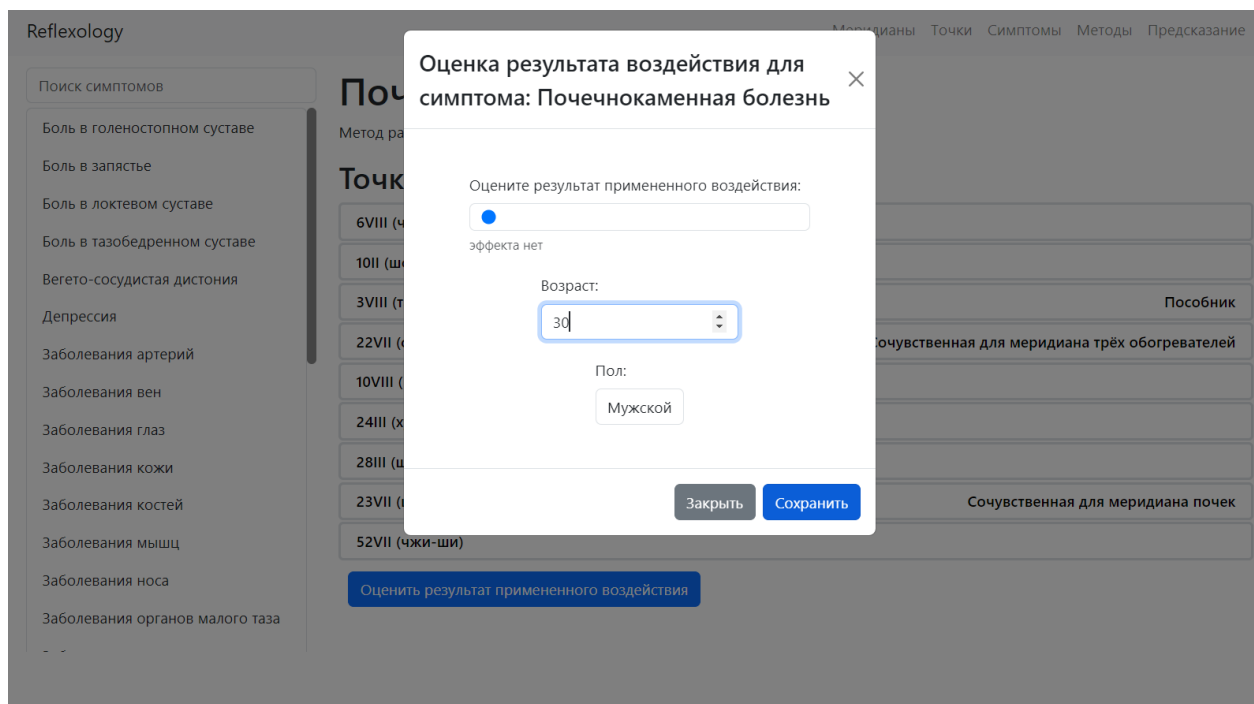


Рисунок 45 – Страница «Симптомы» оценка эффекта

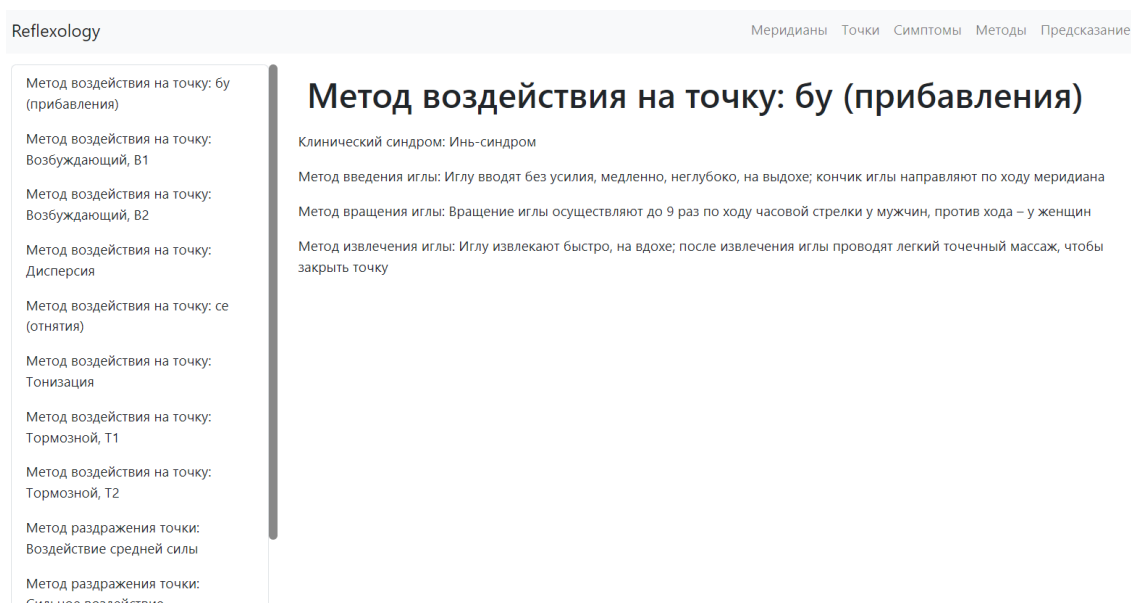


Рисунок 46 – Страница «Методы» при переходе с navbar

Предоставлен список всех методов слева и справа информация, соответствующая методу. Раскрытие описания метода показано на рисунке 47.

Переходя по сущности «Предсказание» показывается форма для заполнения данными по которым пользователь хочет узнать ожидаемый эффект воздействия. Вводятся 3 параметра: симптом, который выбирается из открываемого списка, возраст и пол. Для того чтобы появился результат, нужно нажать на кнопку «Предсказание», которая выделяется голубым цветом.

Страница перехода на «Предсказание» будет выглядеть следующим образом, рисунок 48.

Reflexology

Меридианы Точки Симптомы Методы Предсказание

Метод воздействия на точку: бу (прибавления)

Метод воздействия на точку: Возбуждающий, B1

Метод воздействия на точку: Возбуждающий, B2

Метод воздействия на точку: Дисперсия

Метод воздействия на точку: се (отнятия)

Метод воздействия на точку: Тонизация

Метод воздействия на точку: Тормозной, T1

Метод воздействия на точку: Тормозной, T2

Метод раздражения точки: Воздействие средней силы

Метод раздражения точки: Сильное воздействие

Метод воздействия на точку: Возбуждающий, B1

Глубина введения иглы: 0,5-0,6 сантиметров

Количество точек: 5-10

Время воздействия: 0,5-1 минуту

Характер предусмотренного ощущения: Ощущение укола

Клинический синдром: Инь-синдром

Характеристика клинического синдрома: Обморок, шок, коллапс, потеря сознания

Точки, рекомендуемые для воздействия: Точки скорой помощи, специфические, сегментарные, сигнальные, тонизирующие, ло-пункты

Рисунок 47 – Страница «Методы» при выборе одного из симптомов

Reflexology

Меридианы Точки Симптомы Методы Предсказание

Симптом:

Возраст:

Пол:

Предсказание результата

Рисунок 48 – Страница «Предсказание» при переходе с navbar

Выпадающий список при выборе симптомов продемонстрирован на рисунке 49.

Reflexology

Меридианы Точки Симптомы Методы Предсказание

Симптом:

Вк

- Заболевания глаз
- Эндартериит
- Общая слабость, истощение
- Заболевания носа
- Заболевания ушей
- Заболевания кожи
- Заболевания суставов
- Заболевания костей
- Заболевания мышц
- Заболевания вен
- Заболевания артерий
- Кашель, удушье
- Прединсультное состояние
- Приступ стенокардии
- Лихорадочные заболевания
- Вегето-сосудистая дистония
- Нарушения двигательной функции руки
- Депрессия
- Неврастения
- Заболевания органов малого таза

Рисунок 49 – Страница «Предсказание» выбор симптома из выпадающего списка

Демонстрацию результата предсказания можно увидеть на рисунке 50.

Reflexology

Меридианы Точки Симптомы Методы Предсказание

Симптом: Неврастения

Возраст: 30

Пол: Мужской

Предсказание результата

Предсказание результата воздействия:
крайне-низкий

Рисунок 50 – Страница «Предсказание» отображение результата предсказания

Таким образом, рассмотрены основные сценарии взаимодействия пользователя с базой знаний через систему программных запросов и web-интерфейс.

ЗАКЛЮЧЕНИЕ

Целью данной магистерской диссертации является повышение эффективности работы и оперативной информационной поддержки врача-рефлексотерапевта.

В процессе написания магистерской диссертации была подтверждена актуальность исследуемой проблемы частью которого был анализ предметной области.

В ходе исследования предметной области было подчёркнуто распространение развития и расширения интеллектуальных систем. Рассмотрен принцип работы врача-рефлексотерапевта, для правильной и удобной формализации данных.

Был предложен собственный вариант базы знаний для интеллектуальной системы поддержки врача-рефлексотерапевта.

Проанализированы существующие решения, вывели недостатки, для усовершенствования разработки собственного решения. Основными достоинствами разработанной базы знаний для интеллектуальной системы это доступность, понятная логика использования и полезные информационно-справочные системы, реализующие справочные и обучающие функции для эффективной информационной поддержки врача-рефлексотерапевта.

Для демонстрации возможностей базы знаний для интеллектуальной системы поддержки врача-рефлексотерапевта разработано программное обеспечение с управляющей базой знаний, с организационной системой запросов и визуализацией информационного контента.

На данный момент подсистема экспертных рекомендаций реализована в виде механизма отдельных поисковых запросов. Но имеются перспективы дальнейшего развития, планируется объединить эти механизмы в единую экспертную подсистему с более продвинутым интерфейсом формирования вопросов. Например, вне зависимости от запроса, система выдает комплексные рекомендации. Также есть перспективы развития используемого машинного

обучения, где учитываются смежные заболевания, совместное использование нескольких точек в разных комбинациях, сезон года и время суток, когда проводилось лечение.

Таким образом, требования технического задания магистерской диссертации были выполнены полностью в установленные календарные сроки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Традиционная китайская медицина: сайт. – URL: http://khabarovsk.china-consulate.gov.cn/rus/gyzg/zg7/200405/t20040522_4001536.htm (дата обращения: 16.05.2024).
2. Информационная система: сайт. – URL: https://ru.wikipedia.org/wiki/Информационная_система (дата обращения: 17.05.2024).
3. Shuboshi: Технологии красоты и здоровья: сайт. – URL: <https://shuboshi.com/page/spravochnik-tochek-shuboshi-komfort-atlas> (дата обращения: 18.05.2024).
4. Гаваа Лувсан Очерки методов восточной рефлексотерапии. – 3-е изд., перераб. И доп. -Новосибирск: Наука. Сиб. Отд-ние, 1991. – 432 с.
5. Джованни Мачоча Основы китайской медицины. Подробное руководство для специалистов по акупунктуре и лечению травами. пер. англ. В 3 т. Т. 1. – М.: Рид Элсивер, 2011. – 440 с.
6. Кара-Ушанов В. Ю. SQL-язык реляционных баз данных: учебное пособие. / - Екатеринбург: Изд-во Урал. ун-та, 2016. – 156 с.
7. Кириллов В.В., Громов Г.Ю. Введение в реляционные базы данных. / -СПб.: БХВ-Петербург, 2009. – 464 с.
8. Massage Maps. URL:<https://apps.apple.com/ru/app/massage-maps-%D1%80%D0%B5%D1%84%D0%BB%D0%B5%D0%BA%D1%81%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F/id937929055> (дата обращения 11.10.2023).
9. Foot Reflexology Chart. URL: <https://apps.apple.com/ru/app/foot-reflexology-chart/id929033534> (дата обращения 11.10.2023).
10. Введение в реляционные базы данных / В.В.Кириллов, Г.Ю.Громов. – СПб.: БХВ-Петербург, 2009. – 464 с.

11. Сергеева Т.И. Базы данных: модели данных, проектирование, язык SQL: учеб. пособие / Т.И.Сергеева, М.Ю.Сергеев. Воронеж: ФГБОУ ВПО «Воронежский государственный технический университет», 2012. – 233 с.
12. Гивакс Джей Джей Паттерны проектирования API. – СПб.: Питер, 2023. - 512 с.: ил. – (Серия «Библиотека программиста»).
13. Дронов В. А. React 17. Разработка веб-приложений на JavaScript. – СПб.: БХВ-Петербург, 2022. – 384 с.: ил. – (Профессиональное программирование).
14. Симпсон Кайл Познакомьтесь, JavaScript. 2-е изд. – СПб.: Питер, 2022. – 192 с.: ил. – (Серия «Библиотека программиста»).
15. Касимовская А.Д., Бождай А.С. Разработка базы знаний для интеллектуальной системы поддержки врача-рефлексотерапевта. // НИТиС – 2023: Сборник научных статей по материалам XX Международной научно-технической конференции, посвященной 80-летию юбилею ПГУ. /- Пенза, 2023. - С. 47-51.

ПРИЛОЖЕНИЕ А

(обязательное)

Разработка базы знаний для интеллектуальной системы поддержки врач-
рефлексотерапевта

Листинг программы

ПРИЛОЖЕНИЕ А.1

Файл `index.js`

```
require('dotenv').config()
const express = require('express')
const sequelize = require('./db')
const models = require('./models/models')
const cors = require('cors')
const fileUpload = require('express-fileupload')
const router = require('./routes/index')
const errorHandler =
require('./middleware/ErrorHandlerMiddleware')
const path = require('path')
const fs = require('fs');
const csvWriter = require('csv-write-stream');

const PORT = process.env.PORT || 5000

const app = express()
app.use(cors())
app.use(express.json())
app.use(express.static(path.resolve(__dirname, 'static')))
app.use(fileUpload({}))
app.use('/api', router)

//Обработка ошибок
app.use(errorHandler)

// Установка имени файла для сохранения данных в формате CSV
const filename = 'ratings.csv';

// Создание объекта записи CSV
const writer = csvWriter();
writer.pipe(fs.createWriteStream(filename));
```

```

const fetchDataAndWriteToCSV = async () => {
  try {
    // Получение данных из базы данных
    const ratings = await models.Rating.findAll();
    // Запись данных в CSV файл
    ratings.forEach((rating) => { writer.write({
      symptom_id: rating.symptomId,
      method_id: rating.methodId,
      gender: rating.gender,
      age: rating.age,
      rating: rating.rating_s });
    });

    writer.end();
    console.log('Данные успешно записаны в файл CSV. ');
  } catch (error) {
    console.error('Ошибка при получении данных из
базы данных:', error);
  }
};

const start = async () =>{
  try{
    await sequelize.authenticate()
    await sequelize.sync()
    await fetchDataAndWriteToCSV();
    app.listen(PORT, () => console.log(`Server started on
port ${5000}`))
  } catch(e){
    console.log(e)
  }
}

start()

```

ПРИЛОЖЕНИЕ А.2

Файл db.js

```
const {Sequelize} = require('sequelize');;
require('dotenv').config();
const PORT=5000
const DB_NAME='Reflexology'
const DB_USER='postgres'
const DB_PASSWORD='8'
const DB_HOST='localhost'
const DB_PORT=5433

const sequelize = new Sequelize(
    DB_NAME,
    DB_USER,
    DB_PASSWORD,
    {
        dialect: 'postgres',
        host: DB_HOST,
        port: DB_PORT
    }
);

module.exports = sequelize;
```

ПРИЛОЖЕНИЕ А.3

Файл .env

```
PORT=5000
DB_NAME=Reflexology
DB_USER=postgres
DB_PASSWORD='8'
DB_HOST=localhost
DB_PORT=5433
```

ПРИЛОЖЕНИЕ А.4

Файл indexr.js

```
const Router = require('express')
const router = new Router()
const symptomRouter = require('./symptomRouter')
const ratingRouter = require('./ratingRouter')
const meridianRouter = require('./meridianRouter')
const methodRouter = require('./methodRouter')
const pointRouter = require('./pointRouter')
const symptomPointRouter = require('./symptomPointRouter')

router.use('/symptom', symptomRouter)
router.use('/rating', ratingRouter)
router.use('/point', pointRouter)
router.use('/meridian', meridianRouter)
router.use('/method', methodRouter)
router.use('/symptompont', symptomPointRouter)

module.exports = router
```

ПРИЛОЖЕНИЕ А.5

Файл meridianRouter.js

```
const Router = require('express')
const router = new Router()
const meridianController =
require('../controllers/meridianController')

router.post('/', meridianController.create)
router.get('/', meridianController.getAll)
router.get('/:id', meridianController.getOne)

module.exports = router
```

ПРИЛОЖЕНИЕ A.6

Файл methodRouter.js

```
const Router = require('express')
const router = new Router()
const methodController =
require('../controllers/methodController')

router.post('/',methodController.create)
router.get('/',methodController.getAll)
router.get('/:id',methodController.getOne)

module.exports = router
```

ПРИЛОЖЕНИЕ A.7

Файл pointRouter.js

```
const Router = require('express')
const router = new Router()
const pointController = require('../controllers/pointController')

router.post('/', pointController.create)
router.get('/', pointController.getAll)
router.get('/:id',pointController.getOne)

module.exports = router
```

ПРИЛОЖЕНИЕ A.8

Файл ratingRouter.js

```
const Router = require('express')
const router = new Router()
const ratingController =
require('../controllers/ratingController')

router.post('/', ratingController.create)
router.get('/', ratingController.getAll)
router.get('/:id',ratingController.getOne)
```

```

router.get('/symptoms', ratingController.getAllSymptoms)
router.get('/methods', ratingController.getAllMethods)
router.post('/predict', ratingController.predict)

module.exports = router

```

ПРИЛОЖЕНИЕ A.9

Файл symptomRouter.js

```

const Router = require('express')
const router = new Router()
const symptomController =
require('../controllers/symptomController')

router.post('/', symptomController.create)
router.get('/', symptomController.getAll)
router.get('/:id', symptomController.getOne)

module.exports = router

```

ПРИЛОЖЕНИЕ A.10

Файл models.js

```

const sequelize = require('../db')
const {DataTypes} = require('sequelize')

const Point = sequelize.define('point', {
  id:{type: DataTypes.INTEGER, primaryKey: true, autoIncrement:
true},
  name_p:{type: DataTypes.STRING, unique:true},
  number:{type: DataTypes.STRING, unique:true},
  info:{type: DataTypes.STRING},
  recom:{type: DataTypes.STRING},
  xCoord:{type: DataTypes.INTEGER},
  yCoord:{type: DataTypes.INTEGER},
  location:{type: DataTypes.STRING},
  img:{type: DataTypes.STRING}

```

```

}))

const Symptom = sequelize.define('symptom', {
  id:{type: DataTypes.INTEGER, primaryKey: true, autoIncrement:
true},
  name_s:{type: DataTypes.STRING, unique:true},
  info:{type: DataTypes.STRING},
  pointss:{type: DataTypes.STRING}
}))

const Rating = sequelize.define('rating', {
  rating_s: { type: DataTypes.INTEGER, allowNull: false },
  age: { type: DataTypes.INTEGER },
  gender: { type: DataTypes.INTEGER }
}))

const Method = sequelize.define('method', {
  id:{type: DataTypes.INTEGER, primaryKey: true, autoIncrement:
true},
  name_m:{type: DataTypes.STRING, unique:true},
  info:{type: DataTypes.STRING},
  info2:{type: DataTypes.STRING}
}))

const Meridian = sequelize.define('meridian', {
  id:{type: DataTypes.INTEGER, primaryKey: true, autoIncrement:
true},
  name_m:{type: DataTypes.STRING, unique:true},
  number:{type: DataTypes.STRING, unique:true},
  standart:{type: DataTypes.STRING},
  vse:{type: DataTypes.STRING},
  info:{type: DataTypes.STRING},
  info2:{type: DataTypes.STRING},
  info3:{type: DataTypes.STRING},
  img:{type: DataTypes.STRING}
})

```

```

}))

const SymptomPoint = sequelize.define('symptom_point', {
  id:{type:DataTypes.INTEGER, primaryKey:true,
autoIncrement:true},
}))

Meridian.hasMany(Point)
Point.belongsTo(Meridian)

Method.hasMany(Symptom, {as: 'info_symptom'})
Symptom.belongsTo(Method)

Symptom.hasMany(Rating)
Rating.belongsTo(Symptom)

Method.hasMany(Rating)
Rating.belongsTo(Method)

Symptom.belongsToMany(Point, {through: SymptomPoint})
Point.belongsToMany(Symptom, {through:SymptomPoint })

/*Symptom.hasMany(SymptomPoint)
SymptomPoint.belongsTo(Symptom)

Point.hasMany(SymptomPoint)
SymptomPoint.belongsTo(Point)*/

module.exports = {
  Point,
  Symptom,
  Method,
  Meridian,
  SymptomPoint,
  Rating
}

```

```
}
```

ПРИЛОЖЕНИЕ А.11

Файл meridianController.js

```
const {Meridian} = require('../models/models')
const ApiError = require('../error/ApiError');
const uuid = require('uuid')
const path = require('path');

class MeridianController {

  async create(req,res,next){
    try{
      const{name_m,number, info, info2, info3, vse, standart} =
req.body
      const {img} = req.files
      let fileName = uuid.v4() + ".jpg"
      img.mv(path.resolve(__dirname, '..', 'static', fileName))
      const meridian = await Meridian.create({name_m, number,
info, info2, info3, vse, standart, img:fileName})
      return res.json(meridian)
    } catch (e) {
      next(ApiError.badRequest(e.message))
    }
  }

  async getAll(req, res){
    const meridians = await Meridian.findAll()
    return res.json(meridians)
  }

  async getOne(req,res){
    const {id} = req.params
    const meridian = await Meridian.findOne(
      {
```

```

        where: {id}

      },
    )
    return res.json(meridian)
  }
}

module.exports = new MeridianController()

```

ПРИЛОЖЕНИЕ А.12

Файл methodController.js

```

const {Method, Symptom} = require('../models/models')
const ApiError = require('../error/ApiError');
const uuid = require('uuid')
const path = require('path');

class MethodController {
  async create(req, res, next){
    try{
      let{name_m, info, info2, info_symptom} = req.body
      const method = await Method.create({name_m, info, info2})

      if (info_symptom){
        info_symptom = JSON.parse(info_symptom)
        info_symptom.forEach(i =>
          Symptom.create({
            title: i.title,
            description: i.description,
            methodId: method.id
          })
        )
      }

      return res.json(method)
    }
  }
}

```

```

        } catch (e) {
            next(ApiError.badRequest(e.message))
        }
    }

    async getAll(req, res){
        const methods = await Method.findAll()
        return res.json(methods)
    }

    async getOne(req,res){
        const {id} = req.params
        const method = await Method.findOne(
            {
                where: {id},
                /*include: [{model: Symptom, as:'info_symptom'}]*/
            },
        )
        return res.json(method)
    }
}

module.exports = new MethodController()

```

ПРИЛОЖЕНИЕ А.13

Файл pointController.js

```

const {Point} = require('../models/models')
const ApiError = require('../error/ApiError');
const uuid = require('uuid')
const path = require('path');

class PointController {
    async create(req,res, next){
        try{

```

```

        const{name_p,number, info, recom, xCoord, yCoord,
location} = req.body
        const {img} = req.files
        let fileName = uuid.v4() + ".jpg"
        img.mv(path.resolve(__dirname, '..', 'static', fileName))
        const point = await Point.create({name_p, number, info,
recom, xCoord, yCoord, location,  img: fileName})
        return res.json(point)
    } catch (e) {
        next(ApiError.badRequest(e.message))
    }
}

async getAll(req, res){
    let {meridianId, symptomId, limit, page} = req.query
    page = page || 1
    limit = limit || 100
    let offset = page * limit - limit
    let points;
    if (!meridianId && !symptomId){
        points = await Point.findAndCountAll({limit, offset})
    }
    if(meridianId && !symptomId){
        points = await Point.findAndCountAll(
{where:{meridianId}, limit, offset})
    }
    if(!meridianId && symptomId){
        points = await Point.findAndCountAll(
{where:{symptomId}, limit, offset})
    }
    if(meridianId && symptomId){
        points = await Point.findAndCountAll(
{where:{meridianId, symptomId}, limit, offset})
    }
    return res.json(points)
}

```

```

    }

    async getOne(req, res) {
        const {id} = req.params
        const point = await Point.findOne(
            {
                where: {id}
            },
        )
        return res.json(point)
    }
}

module.exports = new PointController()

```

ПРИЛОЖЕНИЕ А.14

Файл ratingController.js

```

const ApiError = require('../error/ApiError');
const { Rating, Method, Symptom } = require('../models/models');
const { RandomForestClassifier } = require('ml-random-forest');

class ForestModel {
    constructor() {
        this.model = new RandomForestClassifier();
    }

    predict(symptomId, methodId, age, gender) {
        const prediction = this.model.predict([[symptomId,
methodId, age, gender]]);
        return prediction;
    }

    train(features, labels) {
        this.model.train(features, labels);
    }
}

class RatingController {

```

```

constructor() {
  this.forestModel = new ForestModel();
  this.trainModel();
}
async trainModel() {
  try {
    const ratings = await Rating.findAll();
    if (ratings.length === 0) {
      console.log('No ratings found. Skipping model
training.');
```

```

      return;
    }
    const features = ratings.map(rating =>
[rating.symptomId, rating.methodId, rating.age, rating.gender]);
    const labels = ratings.map(rating =>
rating.rating_s);

    this.forestModel.train(features, labels);
    console.log('Model trained successfully');
  } catch (e) {
    console.error('Error training model:', e);
  }
}

async create(req, res, next) {
  try {
    const { rating_s, symptomId, methodId, age, gender }
= req.body;
    const rating = await Rating.create({ rating_s,
symptomId, methodId, age, gender });
    return res.json(rating);
  } catch (e) {
    next(ApiError.badRequest(e.message));
  }
}

```

```

async getAll(req, res) {
  let { symptomId, methodId, limit, page } = req.query;
  page = page || 1;
  limit = limit || 100;
  let offset = page * limit - limit;
  let ratings;
  if (!symptomId && !methodId) {
    ratings = await Rating.findAndCountAll({ limit,
offset });
  } else if (symptomId && !methodId) {
    ratings = await Rating.findAndCountAll({ where: {
symptomId }, limit, offset });
  } else if (!symptomId && methodId) {
    ratings = await Rating.findAndCountAll({ where: {
methodId }, limit, offset });
  } else {
    ratings = await Rating.findAndCountAll({ where: {
symptomId, methodId }, limit, offset });
  }
  return res.json(ratings);
}

async getOne(req, res) {
  const { id } = req.params;
  const rating = await Rating.findOne({ where: { id } });
  return res.json(rating);
}

predict = async (req, res, next) => {
  try {
    //обучение модели при каждом запросе
    await this.trainModel()
    //извлечение данных из тела запроса

```

```

        const { symptomId, methodId, age, gender } =
req.body;

        const method = await Method.findOne({ where: { id:
methodId } });

        const symptom = await Symptom.findOne({ where: { id:
symptomId } });

        if (!method || !symptom) {
            throw new Error('Method or Symptom not found');
        }

        //прогнозирование
        const prediction =
this.forestModel.predict(symptomId, methodId, age, gender);

        return res.json({ prediction });
    } catch (e) {
        next(ApiError.internal(e.message));
    }
}

// метод для получения всех симптомов
async getAllSymptoms(req, res, next) {
    try {
        const symptoms = await Symptom.findAll();
        return res.json(symptoms);
    } catch (e) {
        next(ApiError.internal(e.message));
    }
}

// метод для получения всех методов
async getAllMethods(req, res, next) {
    try {
        const methods = await Method.findAll();
        return res.json(methods);
    } catch (e) {
        next(ApiError.internal(e.message));
    }
}

```

```

    }
  }
}
module.exports = new RatingController();

```

ПРИЛОЖЕНИЕ A.15

Файл symptomController.js

```

const {Symptom} = require('../models/models')
const ApiError = require('../error/ApiError');
const uuid = require('uuid')
const path = require('path');

class SymptomController {
  async create(req, res, next){
    try{
      const{name_s, info, pointss, methodId} = req.body
      const symptom = await Symptom.create({name_s, info,
pointss, methodId})
      return res.json(symptom)
    } catch (e) {
      next(ApiError.badRequest(e.message))
    }
  }

  async getAll(req, res){
    let {methodId, id, limit, page} = req.query
    page = page || 1
    limit = limit || 100
    let offset = page * limit - limit
    let symptoms;
    if (!methodId && !id){
      symptoms= await Symptom.findAndCountAll({limit,
offset})
    }
  }

```

```

        if(methodId && !id){
            symptoms = await Symptom.findAndCountAll(
{where:{methodId}, limit, offset})
        }
        if(!methodId && id){
            symptoms = await Symptom.findAndCountAll(
{where:{id}, limit, offset})
        }
        if(methodId && id){
            symptoms = await Symptom.findAndCountAll(
{where:{methodId, id}, limit, offset})
        }
        return res.json(symptoms)
    }

    async getOne(req,res){
        const {id} = req.params
        const symptom = await Symptom.findOne(
            {
                where: {id}
            },
        )
        return res.json(symptom)
    }
}

module.exports = new SymptomController()

```

ПРИЛОЖЕНИЕ А.16

Файл site.js

```

import {makeAutoObservable, makeObservable} from "mobx";

export default class Site {
    constructor() {
        this._meridians= [

```

```

    ]
    this._symptoms = [
    ]
    this._points = [
    ]
    this._methods = [
    ]
    this._ratings = [
    ]

    this._selectedMethod = {}
    this._selectedMeridian = {}
    this._selectedSymptom = {}
    this._selectedRating = {}
    this._selectedPoint = {}
    makeAutoObservable(this)
}

setMeridians(meridians) {
    this._meridians = meridians
}
setPoints(points) {
    this._points = points
}
setMethods(methods) {
    this._methods = methods
}
setSymptoms(symptoms) {
    this._symptoms = symptoms
}
setRatings(ratings) {
    this._ratings = ratings
}

```

```

setSelectedSymptom(symptoms) {
    this._selectedSymptom = symptoms
}
setSelectedMethod(methods) {
    this._selectedMethod = methods
}
setSelectedMeridian(meridians) {
    this._selectedMeridian = meridians
}
setSelectedPoint(points) {
    this._selectedPoint = points
}

setSelectedRating(ratings) {
    this._selectedRating = ratings
}

get meridians() {
    return this._meridians
}
get points() {
    return this._points
}
get methods() {
    return this._methods
}
get symptoms() {
    return this._symptoms
}
get ratings() {
    return this._ratings
}
get selectedSymptom() {
    return this._selectedSymptom
}

```

```

get selectedRating() {
    return this._selectedRating
}

get selectedMethod() {
    return this._selectedMethod
}

get selectedMeridian() {
    return this._selectedMeridian
}

get selectedPoint() {
    return this._selectedPoint
}
}

```

ПРИЛОЖЕНИЕ А.17

Файл routes.js

```

import { Component } from "react";
import Main from "../pages/main";
import Point from "../pages/point";
import Symptom from "../pages/symptom";
import Meridian from "../pages/meridian";
import Method from "../pages/method";
import PredictionForm from "../pages/PredictionForm";
import { MAIN_ROUTE, POINT_ROUTE, SYMPTOM_ROUTE,
MERIDIAN_ROUTE, METHOD_ROUTE, PREDICTION_ROUTE } from
"./utils/consts";

export const publicRoutes = [
    { path: MAIN_ROUTE,
      Component: Main
    },
    { path: POINT_ROUTE,
      Component: Point
    }
]

```

```

    },
    { path: SYMPTOM_ROUTE,
      Component: Symptom
    },
    { path: MERIDIAN_ROUTE,
      Component: Meridian
    },
    { path: METHOD_ROUTE,
      Component: Method
    },
    { path: PREDICTION_ROUTE,
      Component: PredictionForm
    },
  ]

```

ПРИЛОЖЕНИЕ A.18

Файл NavBar.js

```

import React from "react";
import Container from 'react-bootstrap/Container';
import Nav from 'react-bootstrap/Nav';
import Navbar from 'react-bootstrap/Navbar';
import NavDropdown from 'react-bootstrap/NavDropdown';
import SymptomList from "../pages/symptomList";
import 'bootstrap/dist/css/bootstrap.min.css';

function NavBar() {
  return (
    <Navbar expand="lg" className="navbar fixed-top navbar-light
bg-light" >
      <Container>
        <Navbar.Brand href="/main">Reflexology</Navbar.Brand>
        <Navbar.Toggle aria-controls="basic-navbar-nav" />
        <Navbar.Collapse id="basic-navbar-nav">
          <Nav className="ml-auto">
            <Nav.Link href="/meridian">Меридианы</Nav.Link>

```

```

        <Nav.Link href="/point">Точки</Nav.Link>
        <Nav.Link href="/symptom">Симптомы</Nav.Link>
        <Nav.Link href="/method">Методы</Nav.Link>
        <Nav.Link href="/predict">Предсказание</Nav.Link>
    </Nav>
  </Navbar.Collapse>
</Container>
</Navbar>
);
}
export default NavBar;

```

ПРИЛОЖЕНИЕ А.19

Файл index.js

```

import React, { createContext } from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import Site from './site';

export const Context = createContext(null)

const root =
ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <Context.Provider value={{
    site: new Site()}}>
    <App />
  </Context.Provider>,
);

reportWebVitals();

```

ПРИЛОЖЕНИЕ A.20

Файл AppRouter.js

```
import React from 'react';
import { BrowserRouter } from 'react-router-dom';
import { Routes, Route, Navigate } from 'react-router-dom'
import Main from './pages/main'
import { publicRoutes } from './routes';

const AppRouter = () => {
  return(
    <Routes>
      {publicRoutes.map(({path, Component}) =>
        <Route key={path} path={path} element={<Component/>}
exact/>
      )}
      <Route path="*" element={<Main/>}/>
    </Routes>
  );
};

export default AppRouter;
```

ПРИЛОЖЕНИЕ A.21

Файл App.js

```
import { BrowserRouter } from 'react-router-dom';
import './App.css';
import AppRouter from './AppRouter';
import NavBar from './NavBar';
import Main from './pages/main';
import { observer } from 'mobx-react-lite';
import { useEffect, useState } from 'react';

const App = () => {
  return (
```

```

    <BrowserRouter >
      <NavBar/>
      <AppRouter />
    </BrowserRouter>
  );
};
export default App;

```

ПРИЛОЖЕНИЕ A.22

Файл symptomModal.js

```

import React, { useState } from 'react';
import { Modal, Button, Form } from 'react-bootstrap';

const SymptomModal = ({ show, handleClose, symptom, methodId,
handleSave }) => {
  const [rating_s, setRating] = useState(1);
  const [age, setAge] = useState('');
  const [gender, setGender] = useState(1);
  const [ageError, setAgeError] = useState('');

  const handleRatingChange = (e) => { setRating(e.target.value);
};

  const handleAgeChange = (e) => {
    setAge(e.target.value);
    if (e.target.value) {
      setAgeError('');
    }
  };

  const handleGenderChange = (e) => { setGender(e.target.value);
};

  const saveRating = () => {
    if (!age) {
      setAgeError('Пожалуйста, укажите возраст');

```

```

        return;
    }
    if (symptom && symptom.id && methodId) {
        handleSave(symptom.id, methodId, rating_s, age, gender);
        handleClose();
    } else {
        console.error('Ошибка: symptom, symptom.id или methodId не
определены');
    }
};

const getRatingDescription = (rating) => {
    switch (parseInt(rating)) {
        case 1: return 'эффекта нет';
        case 2: return 'крайне-низкий';
        case 3: return 'низкий';
        case 4: return 'средний';
        case 5: return 'высокий';
        case 6: return 'максимальный';
        default: return '';
    }
};

return (
    <Modal show={show} onHide={handleClose}>
        <Modal.Header closeButton>
            <Modal.Title>Оценка результата воздействия для симптома:
{symptom.name_s}</Modal.Title>
        </Modal.Header>
        <Modal.Body>
            <Form>
                <Form.Group controlId="formBasicRange">
                    <Form.Label>Оцените результат примененного
воздействия:</Form.Label>
                    <Form.Control

```

```

        type="range"
        min="1"
        max="6"
        value={rating_s}
        onChange={handleRatingChange}
      />
      <Form.Text className="text-
muted">{getRatingDescription(rating_s)}</Form.Text>
    </Form.Group>
    <Form.Group controlId="formAge">
      <Form.Label>Возраст:</Form.Label>
      <Form.Control
        type="number"
        value={age}
        onChange={handleAgeChange}
      />
      {ageError && <Form.Text className="text-
danger">{ageError}</Form.Text>}}
    </Form.Group>
    <Form.Group controlId="formGender">
      <Form.Label>Пол:</Form.Label>
      <Form.Control
        as="select"
        value={gender}
        onChange={handleGenderChange}
      >
        <option value={1}>Мужской</option>
        <option value={0}>Женский</option>
      </Form.Control>
    </Form.Group>
  </Form>
</Modal.Body>
<Modal.Footer>
  <Button variant="secondary" onClick={handleClose}>
    Закрыть

```

```

        </Button>
        <Button variant="primary" onClick={saveRating}>
            Сохранить
        </Button>
    </Modal.Footer>
</Modal>
);
};

export default SymptomModal;

```

ПРИЛОЖЕНИЕ А.23

Файл symptomList.js

```

import React, { useState } from "react";
import ListGroup from 'react-bootstrap/ListGroup';
import { Context } from "../index";
import { useContext } from "react";

const SymptomList = ({onSelectSymptom}) => {
    const {site} = useContext(Context);
    const [searchQuery, setSearchQuery] = useState("");
    const handleSelectSymptom = symptom => {
        onSelectSymptom(symptom.id);
        site.setSelectedSymptom(symptom);
    };

    const filteredSymptoms = site.symptoms.filter(symptom =>
        symptom.name_s.toLowerCase().includes(searchQuery.toLowerCase()
    )) ||
    (symptom.info &&
        symptom.info.toLowerCase().includes(searchQuery.toLowerCase())) ||
    (symptom.pointss &&
        symptom.pointss.toLowerCase().includes(searchQuery.toLowerCase()
    ));

```

```

    const sortedSymptom = filteredSymptoms.sort((a,b) =>
a.name_s.localeCompare(b.name_s));

    if (!site || !Array.isArray(site.symptoms)) {
        return null; // or a loading spinner or error message
    }

    return(
<div >
<input type="text" className="form-control"
style={{ marginBottom:5}} placeholder="Поиск симптомов"
value={searchQuery} onChange={e =>
setSearchQuery(e.target.value)} />
    <div style={{ height: "575px", overflowY: "auto" }}
className="custom-scrollbar">
        <ListGroup >
            {sortedSymptom.map(symptom=>(
                <ListGroup.Item
                    style={{cursor: 'pointer'}}
                    active ={symptom.id === site.selectedSymptom.id}
                    onClick={() => handleSelectSymptom(symptom)}
                    key={symptom.id}>
                        {symptom.name_s}
                </ListGroup.Item>))}
        </ListGroup>
    </div>
</div>
);
};

export default SymptomList;

```

ПРИЛОЖЕНИЕ А.24

Файл symptomDescription.js

```

import React, { useEffect, useState } from "react";
import axios from "axios";
import SymptomModal from "../symptomModal";
import { Button } from "react-bootstrap";
import {$host} from "../http/index.js";

const SymptomDescription = ({ diseaseId }) => {
  const [site, setSymptomData] = useState(null);
  const [methodData, setMethodData] = useState(null);
  const [spData, setSPData] = useState([]);
  const [selectedPointIndex, setSelectedPointIndex] =
    useState(null);
  const [modalShow, setModalShow] = useState(false);
  const [selectedSymptom, setSelectedSymptom] = useState(null);
  const [selectedMethodId, setSelectedMethodId] = useState(null);

  if (methodData !== null){
    var infoSplitList = methodData.info2.split('$');
  }

  useEffect(() => {
    setSelectedPointIndex(null);
    const fetchSymptomData = async () => {
      try {
        const response = await
          axios.get(`http://localhost:5000/api/symptom/${diseaseId}`);
        setSymptomData(response.data);

        const methodResponse = await
          axios.get(`http://localhost:5000/api/method/${response.data.metho
            dId}`);
        setMethodData(methodResponse.data);
      }
    }
  })
}

```

```

        const spResponse = await
axios.get(`http://localhost:5000/api/symptompoint?symptomId=${diseaseId}`);

        const pointsData = await
Promise.all(spResponse.data.rows.map(async (symptomPoint) => {
            const pointResponse = await
axios.get(`http://localhost:5000/api/point/${symptomPoint.pointId}`);

            return pointResponse.data;
        }));
        setSPData(pointsData);
    } catch (error) {
        console.error('Ошибка при загрузке данных:', error);
    }
};

    if (diseaseId) { fetchSymptomData(); }
}, [diseaseId]);

const handleClick = () => {
    setMethodData(prevMethodData => ({
        ...prevMethodData,
        showInfo2: !prevMethodData.showInfo2
    }));
};

const handleClickPoint = (index) => {
    if (selectedPointIndex === index) {
        setSelectedPointIndex(null);
    } else {
        setSelectedPointIndex(index);
    }
};

```

```

const handleOpenModal = (symptom) => {
  if (symptom) {
    setSelectedMethodId(symptom.methodId); // Сохраняем ID
метода
    setSelectedSymptom(symptom);
    setModalShow(true);
  } else {
    console.error('Ошибка: symptom не определен');
  }
};

const saveRatingToServer = async (symptomId, methodId,
rating_s, age, gender) => {
  try {
    await $host.post('api/rating', { symptomId, methodId,
rating_s, age, gender });
  } catch (error) {
    console.error('Ошибка при сохранении оценки:', error);
  }
};

const handleSaveRating = async (symptomId, methodId, rating_s,
age, gender) => {
  await saveRatingToServer(symptomId, methodId, rating_s, age,
gender);
};

return (
  <div>
    {site ? (
      <div>
        <h1>{site.name_s}</h1>
        {methodData && (
          <div>

```

```

        <p onClick={handleClick} style={{ cursor:
'pointer', textDecoration: 'none', borderBottom: '1px solid
transparent' }}
            onMouseEnter={(e) => e.target.style.borderBottom
= '1px solid black'}
            onMouseLeave={(e) => e.target.style.borderBottom
= '1px solid transparent'}}>
            {methodData.name_m}
        </p>
        {methodData.showInfo2 && <p>{infoSplitList.map(item
=><p>{item}</p>)}</p>}}
    </div>
  )}

  <h2>Точки симптома:</h2>
  <div>
    {spData && spData.length > 0 ? (
      <div>
        {spData.map((item, index) => (
          <div key={index}>
            <h6 onClick={() => handleClickPoint(index)} style={{
              cursor: "pointer",
              backgroundColor: selectedPointIndex === index ? "#007bff" :
"transparent",
              color: selectedPointIndex === index ? "#ffffff" :
"#000000",
              border: "3px solid #dee2e6",
              borderRadius: "5px",
              marginBottom: "2px",
              padding: "0.5rem 1rem"
            }}
            className="list-group-item d-flex justify-content-between
align-items-center">
              <span>{item.number} ({item.name_p}) </span>
              <span>{item.info}</span>

```

```

    </h6>
  </div>
  )})

    </div>
  ) : (
    <p>Нет данных о точках симптома.</p>
  )}
</div>
</div>
) : (
  <div>

    <h4 style={{padding:10, fontFamily:'Lato', textAlign:
'center', color:'red'}}>В этих случаях взаимодействовать на точки
нельзя, и иного варианта быть не может! </h4>

    <h5 style={{padding:20, fontFamily:'Lato', textAlign:
'center'}}>
      Противопоказания: беременность, период обострения
хронических заболеваний, признаки ОРВИ, инфекции и онкологические
заболевания.

    </h5>
  </div>
)}

{selectedPointIndex !== null && (
  <div>
    <h3>Описание точки: </h3>
    <div dangerouslySetInnerHTML={{ __html: `Номер:
${spData[selectedPointIndex].number} <br>
${spData[selectedPointIndex].recom} <br><br> Расположение:
${spData[selectedPointIndex].location} ` }} />
    <img src={ process.env.REACT_APP_API_URL +
spData[selectedPointIndex].img} alt={`Изображение точки
${spData[selectedPointIndex].number}`} />
  </div>

```

```

    })
    <div> <p style={{padding:10}}>
      {site && <Button onClick={() =>
handleOpenModal(site)}>Оценить результат примененного
воздействия</Button>}
      {selectedSymptom && (
        <SymptomModal
          show={modalShow}
          handleClose={() => setModalShow(false)}
          symptom={selectedSymptom}
          methodId={selectedMethodId}
          handleSave={handleSaveRating}
        />
      )}
    </p>
  </div>
</div>
);
};

export default SymptomDescription;

```

ПРИЛОЖЕНИЕ А.25

Файл symptom.js

```

import React, { useContext, useEffect, useState } from "react";
import { Container } from "react-bootstrap";
import Row from "react-bootstrap/Row";
import Col from "react-bootstrap/Col";
import SymptomList from './symptomList';
import SymptomDescription from './symptomDescription';
import { Context } from "../index";
import { fetchSymptom } from "../http/siteAPI";
import { observer } from "mobx-react-lite";

```

```

const Symptom = observer(() => {
  const {site} = useContext(Context)
  const [diseaseId, setDiseaseId] = useState(null)
  const [symptomLoaded, setSymptomsLoaded] = useState(false);

  useEffect(()=>{
    fetchSymptom().then(data =>
{site.setSymptoms(data);  setSymptomsLoaded(true)} )
    , [])

    return(
      <div style={{padding:60,}}>
        <Container>
          <Row className="mt-2">
            <Col md='3'> {symptomLoaded && <SymptomList
onSelectSymptom={setDiseaseId}/>}</Col>
            <Col md='9'><SymptomDescription
diseaseId={diseaseId}/></Col>
          </Row>
        </Container>
      </div>
    );
  });

export default Symptom;

```

ПРИЛОЖЕНИЕ А.26

Файл PredictionForm.js

```

import React, { useState, useEffect } from 'react';
import { $host } from "../http/index.js";

const PredictionForm = () => {
  const [symptomId, setSymptomId] = useState(5);

```

```

const [age, setAge] = useState('');
const [gender, setGender] = useState(1);
const [prediction, setPrediction] = useState(null);
const [symptoms, setSymptoms] = useState([]);
const [loading, setLoading] = useState(true);
const [error, setError] = useState(null);
const defaultMethodId = 9; // Значение метода по умолчанию

useEffect(() => {
  const fetchSymptoms = async () => {
    try {
      const response = await $host.get('api/symptom');
      setSymptoms(response.data.rows || []);
    } catch (error) {
      console.error('There was an error fetching the
symptoms!', error);
      setError('There was an error fetching the
symptoms.');
```

```

        symptomId: parseInt(symptomId),
        methodId: defaultMethodId, // Устанавливаем
значение метода по умолчанию
        age: parseInt(age),
        gender: parseInt(gender)
    });
    setPrediction(response.data.prediction);
} catch (error) {
    console.error('There was an error making the
prediction request!', error);
    setError('There was an error making the prediction
request.');
```

}

```

};

const getPredictionDescription = (prediction) => {
    switch (parseInt(prediction)) {
        case 1: return 'эффекта нет';
        case 2: return 'крайне-низкий';
        case 3: return 'низкий';
        case 4: return 'средний';
        case 5: return 'высокий';
        case 6: return 'максимальный';
        default: return '';
    }
};

if (loading) {
    return <p>Loading...</p>;
}

if (error) {
    return <p>{error}</p>;
}

```

```

return (
  <div>
    <h2>Rating Prediction</h2>
    <form onSubmit={handleSubmit} >
      <div>
        <label>Симптом:</label>
        <select
          value={symptomId}
          onChange={ (e) =>
setSymptomId(e.target.value)} required>
          {symptoms.map(symptom => (
            <option key={symptom.id}
value={symptom.id}>
              {symptom.name_s}
            </option>
          ))}
        </select>
      </div>
      <div>
        <label>Возраст:</label>
        <input
          type="number"
          value={age}
          onChange={ (e) => setAge(e.target.value)}
          required
        />
      </div>
      <div>
        <label>Пол:</label>
        <select value={gender}
          onChange={ (e) =>
setGender(e.target.value)} required>
          <option value={1}>Мужской</option>
          <option value={0}>Женский</option>
        </select>

```

```

        </div>
        <button type="submit">Предсказание
результата</button>
    </form>
    {prediction !== null && (
        <div className='centerDiv'>
            <h3>Предсказание результата воздействия:</h3>
            <h2 style={{ color: 'red'
}}>{getPredictionDescription(prediction)}</h2>
        </div>
    )}
</div>
);
};

export default PredictionForm;

```

ПРИЛОЖЕНИЕ А.27

Файл pointList.js

```

import React, { useState } from "react";
import ListGroup from 'react-bootstrap/ListGroup';
import { Context } from "../index";
import { useContext } from "react";

const PointList = ({onSelectPoint}) => {
    const {site} = useContext(Context);
    const [searchQuery, setSearchQuery] = useState("");
    const handleSelectPoint = point => {
        onSelectPoint(point.id);
        site.setSelectedPoint(point);
    };

    const filteredPoints = site.points.filter(point =>

```

```

        point.name_p.toLowerCase().includes(searchQuery.toLowerCase
    )) ||
    (point.info &&
point.info.toLowerCase().includes(searchQuery.toLowerCase())) ||
    (point.number &&
point.number.toLowerCase().includes(searchQuery.toLowerCase())));
    const sortedPoint = filteredPoints.sort((a,b) =>
a.name_p.localeCompare(b.name_p));

    if (!site || !Array.isArray(site.points)) {
        return null; // or a loading spinner or error message
    }

    return(
<div >
<input type="text" class="form-control" style={{
marginBottom:5}} placeholder="Поиск точек" value={searchQuery}
    onChange={e => setSearchQuery(e.target.value)} />
    <div style={{ height: "575px", overflowY: "auto" }}
className="custom-scrollbar">
        <ListGroup >
            {sortedPoint.map(point=>(
                <ListGroup.Item
                    style={{cursor: 'pointer'}}
                    active ={point.id === site.selectedPoint.id}
                    onClick={() => handleSelectPoint(point)}
                    key={point.id}>
                        {point.name_p} ({point.number})
                </ListGroup.Item>))}
        </ListGroup>
    </div>
</div>
);};

export default PointList;

```

ПРИЛОЖЕНИЕ А.28

Файл pointDescription.js

```
import React, { useEffect, useState } from "react";
import axios from "axios";
import Col from "react-bootstrap/Col";
import Image from "react-bootstrap/Image";
import { Container } from "react-bootstrap";
import Row from "react-bootstrap/Row";
import yn from './yn.jpg';

const PointDescription = ({diseaseId}) => {
  const [site, setPointData] = useState(null)
  const [meridianData, setMeridianData] = useState(null);

  useEffect(() => {
    const fetchPointData = async () => {
      try {
        const response = await
axios.get(`http://localhost:5000/api/point/${diseaseId}`);
        setPointData(response.data);

        const meridianResponse = await
axios.get(`http://localhost:5000/api/meridian/${response.data.mer
idianId}`);
        setMeridianData(meridianResponse.data);

      } catch (error) {
        console.error('Ошибка при загрузке данных:', error);
      }
    };

    if(diseaseId){
      fetchPointData();
    }
  });
}
```

```

    }
  }, [diseaseId]);

const handleClick = () => {
  setMeridianData(prevMeridianData => ({
    ...prevMeridianData,
    showStandart: !prevMeridianData.showStandart
  }));
};

return (
  <div>
    {site ? (
      <Container>
        <h1>{site.name_p}</h1>
        <p>Homep: {site.number}</p>
        {meridianData &&
          <div>
            <p onClick={handleClick} style={{ cursor: 'pointer',
textDecoration: 'none', borderBottom: '1px solid transparent' }}
            onMouseEnter={(e) => e.target.style.borderBottom = '1px solid
black'}
            onMouseLeave={(e) => e.target.style.borderBottom = '1px solid
transparent'}}>
              Точка входит в {meridianData && meridianData.name_m}
            </p>
            {meridianData.showStandart &&
              <p>{meridianData.standart.split('$')}</p>
            }
          </div>
        }
        <h6>{site.info}</h6>
        <p>Расположение: {site.location}</p>

        <Row className="mt-1">
          <Col md='6'>

```

```

      <Image src={ process.env.REACT_APP_API_URL +
site.img}/>
    </Col>
    <Col md='6'>

    </Col>
  </Row>
</Container>
) : (
  <div>
    <h4 style={{padding:20, fontFamily:'Lato'}} >
Акупунктурная точка - это участок кожи и подкожной основы,
который имеет комплекс взаимосвязанных микроструктур
      (кровеносных сосудов, нервов, клеток соединительной
ткани и др.), создающих активную зону. <br/>
      Эта зона оказывает влияние на нервные окончания и
осуществляет связь между участком
      кожи и внутренним органом. Область точки
характеризуется усиленным поглощением кислорода,
      более высокой по сравнению с рядом лежащими
участками кожи температурой, более низким
      электрическим сопротивлением.</h4>
    <h5 style={{fontFamily:'Open Sans'}}>
Признаком правильного нахождения точки акупунктуры
является появление при надавливании на
      область ее расположения особых ощущений, которые
называются предусмотренными. Это те ощущения,
      которые ожидает получить врач при воздействии на
точку. Пациент отмечает появление в области точки
      ощущения ломоты, тяжести, распираия, болезненности,
онемения, укола, о чем он и сообщает врачу.
      В зависимости от клинической картины заболевания
врач ставит задачу добиться сильных, слабых
предусмотренных ощущений или ощущений средней силы.

```

Иногда при воздействии на точку возникает иррадиация возникшего ощущения вверх, вниз, в сторону.

</h5>

<h4 style={{padding:20, fontFamily:'Open Sans'}}>Расположение каждой точки принято искать через признаки тела и расстояние по цуням. Так выглядят цуни:</h4>

<h5 style={{fontFamily:'Open Sans'}} > У каждой точки прописаны цифро-цифрового обозначения.

Такое освоение точек следует признать более простым и доступным.

Использование цифро-цифрового обозначения точек при составлении методических пособий и атласов

в значительной степени облегчает процесс обучения врачей, впервые осваивающих метод рефлексотерапии.

Такое обозначение точек акупунктуры является целесообразным в методической литературе, рассчитанной на широкий круг читателей, и способствует распространению применения данного метода среди населения.

</h5>

</div>

)}

</div>

);

};

export default PointDescription;

ПРИЛОЖЕНИЕ А.29

Файл point.js

```
import React, { useContext, useEffect, useState } from "react";
import { Container } from "react-bootstrap";
import Row from "react-bootstrap/Row";
import Col from "react-bootstrap/Col";
```

```

import PointList from './pointList';
import PointDescription from './pointDescription';
import { Context } from "../index";
import { fetchPoint } from "../http/siteAPI";
import { observer } from "mobx-react-lite";

const Point = observer(() => {
  const {site} = useContext(Context)
  const [diseaseId, setDiseaseId] = useState(null)
  const [pointLoaded, setPointsLoaded] = useState(false);

  useEffect(()=>{
    fetchPoint().then(data =>
{site.setPoints(data);  setPointsLoaded(true)} )
    , [])

    return(
      <div style={{padding:60,}}>
        <Container>
          <Row className="mt-2">
            <Col md='3'> {pointLoaded && <PointList
onSelectPoint={setDiseaseId}/>}</Col>
            <Col md='9'><PointDescription
diseaseId={diseaseId}/></Col>
          </Row>
        </Container>
      </div> );
  });

export default Point;

```

ПРИЛОЖЕНИЕ А.30

Файл methodList.js

```

import React from "react";

```

```

import ListGroup from 'react-bootstrap/ListGroup';
import { Context } from "../index";
import { useContext } from "react";

const MethodList = ({onSelectMethod}) => {
  const {site} = useContext(Context);
  const handleSelectMethod = method => {
    onSelectMethod(method.id);
    site.setSelectedMethod(method);};

  if (!site || !Array.isArray(site.methods)) {
    return null; // or a loading spinner or error message
  }

  return(
    <div style={{ height: "620px", overflowY: "auto" }}
    className="custom-scrollbar">
      <ListGroup >
        {site.methods.sort((a,b) =>
a.name_m.localeCompare(b.name_m)).map(method=>(
          <ListGroup.Item
            style={{cursor: 'pointer'}}
            active={method.id === site.selectedMethod.id}
            onClick={() => handleSelectMethod(method)}
            key={method.id}>
              {method.name_m}
            </ListGroup.Item>))}
        </ListGroup>
      </div>
    );
  };
};

export default MethodList ;

```

ПРИЛОЖЕНИЕ А.31

Файл methodDescription.js

```
import React, { useEffect, useState } from "react";
import axios from "axios";

const MethodDescription = ({diseaseId}) => {

  const [site, setMethodData] = useState(null)

  if (site !== null){
    var infoSplitList = site.info2.split('$');
  }

  useEffect(() => {
    const fetchMethodData = async () => {
      try {
        const response = await
axios.get(`http://localhost:5000/api/method/${diseaseId}`);
        setMethodData(response.data);
      } catch (error) {
        console.error('Ошибка при загрузке данных:', error);
      }
    };

    if(diseaseId){
      fetchMethodData();
    }
  }, [diseaseId]);

  return (
    <div>
      {site ? (
        <div>
          <h1 style={{padding:10}}>{site.name_m}</h1>

```

```

        {infoSplitList.map(item =><p>{item}</p>)}
      </div>
    ) : (
      <h4>Выбери метод...</h4>
    )}
  </div>
);
};

```

```
export default MethodDescription;
```

ПРИЛОЖЕНИЕ А.32

Файл method.js

```

import React, { useContext, useEffect, useState } from "react";
import { Container } from "react-bootstrap";
import Row from "react-bootstrap/Row";
import Col from "react-bootstrap/Col";
import MethodList from './methodList';
import MethodDescription from './methodDescription';
import { Context } from "../index";
import { fetchMethod } from "../http/siteAPI";
import { observer } from "mobx-react-lite";

const Method = observer(() => {
  const {site} = useContext(Context)
  const [diseaseId, setDiseaseId] = useState(12)
  const [methodLoaded, setMethodsLoaded] = useState(false);

  useEffect(()=>{
    fetchMethod().then(data =>
{site.setMethods(data); setMethodsLoaded(true)} )
  }, [])

  return(

```

```

<div style={{padding:60,}}>
  <Container>
    <Row className="mt-2">
      <Col md='3'> {methodLoaded && <MethodList
onSelectMethod={setDiseaseId}/>}</Col>
      <Col md='9'><MethodDescription
diseaseId={diseaseId}/></Col>
    </Row>
  </Container>
</div>
);
});

export default Method;

```

ПРИЛОЖЕНИЕ А.33

Файл meridianList.js

```

import React, { useState } from "react";
import ListGroup from 'react-bootstrap/ListGroup';
import { Context } from "../index";
import { useContext } from "react";

const MeridianList = ({onSelectMeridian}) => {
  const {site} = useContext(Context);
  const [searchQuery, setSearchQuery] = useState("");
  const handleSelectMeridian = meridian => {
    onSelectMeridian(meridian.id);
    site.setSelectedMeridian(meridian); };

  const filteredMeridians = site.meridians.filter(meridian =>
    meridian.name_m.toLowerCase().includes(searchQuery.toLowerCase()
ase())) ||

```

```

    (meridian.number &&
meridian.number.toLowerCase().includes(searchQuery.toLowerCase()))
)||
    (meridian.info3 &&
meridian.info3.toLowerCase().includes(searchQuery.toLowerCase()))
||
    (meridian.info2 &&
meridian.info2.toLowerCase().includes(searchQuery.toLowerCase()))
||
    (meridian.info &&
meridian.info.toLowerCase().includes(searchQuery.toLowerCase())) |
|
    (meridian.vse &&
meridian.vse.toLowerCase().includes(searchQuery.toLowerCase()))
    .sort((a,b) => a.id - b.id);

    if (!site || !Array.isArray(site.meridians)) {
        return null; // or a loading spinner or error message
    }

    return(
        <div>
<input type="text" class="form-control" style={{
marginBottom:5}} placeholder="Поиск меридианов"
value={searchQuery}
    onChange={e => setSearchQuery(e.target.value)} />

<ListGroup >
    {filteredMeridians.map(meridian=>(
        <ListGroup.Item
            style={{cursor: 'pointer'}}
            active ={meridian.id === site.selectedMeridian.id}
            onClick={ () => handleSelectMeridian(meridian) }
            key={meridian.id}>
            {meridian.name_m}

```

```

        </ListGroup.Item>)) }
    </ListGroup>
  </div>
);
};

export default MeridianList ;

```

ПРИЛОЖЕНИЕ А.34

Файл meridianDescription.js

```

import React, { useEffect, useState } from "react";
import axios from "axios";
import { Container } from "react-bootstrap";
import Row from "react-bootstrap/Row";
import Col from "react-bootstrap/Col";
import Image from "react-bootstrap/Image";
import otnoshenie from './otnoshenie.png';

const MeridianDescription = ({diseaseId}) => {

  const [site, setMeridianData] = useState(null)
  const [pointData, setPointData] = useState(null);
  const [selectedPoint, setSelectedPoint] = useState(null);

  if (site !== null){
    var infoSplitList = site.standart.split('$');
  }

  useEffect(() => {
    setSelectedPoint(null);
    const fetchMeridianData = async () => {
      try {
        const response = await
        axios.get(`http://localhost:5000/api/meridian/${diseaseId}`);
        setMeridianData(response.data);

```

```

        const pointResponse = await
axios.get(`http://localhost:5000/api/point?meridianId=${diseaseId
}`);

        setPointData(pointResponse.data.rows);


    } catch (error) {
        console.error('Ошибка при загрузке данных:', error);
    }
};


    if(diseaseId){
        fetchMeridianData();
    }
}, [diseaseId]);

const handlePointClick = (point) => {
    if(selectedPoint && selectedPoint.id === point.id){
        setSelectedPoint(null);
    } else{
        setSelectedPoint(point);
    }
};

return (
    <div>
        {site ? (
            <Container>
                <h1>{site.name_m}</h1>
                <p>{site.number}</p>
                <Row className="mt-1">
                    <Col md='5'>

```

```

<Image src={ process.env.REACT_APP_API_URL +
site.img}/>

{selectedPoint && (
  <div
    className="selected-point-marker"
    style={{
      position: "absolute",
      top: selectedPoint.yCoord,
      left: selectedPoint.xCoord,
      width: "10px",
      height: "10px",
      backgroundColor: "blue",
      borderRadius: "50%",
      animation: "blinking 1s infinite",
    }}
  ></div>
)}
</Col>
<Col md='7'>
<h3>Точки меридиана:</h3>
<div>
  {pointData && pointData
.sort((a, b) => parseInt(a.number) - parseInt(b.number))
.map((point) => (
    <li key={point.id}
      onClick={() => handlePointClick(point)}
      style={{ cursor: "pointer", backgroundColor:
selectedPoint === point ? "#007bff" : "transparent", color:
selectedPoint === point ? "#ffffff" : "#000000", border: "3px
solid #dee2e6", borderRadius: "5px", marginBottom: "2px",
padding: "0.5rem 1rem" }}
      className="list-group-item d-flex justify-content-between
align-items-center"
    >

```

```

        <span>{point.number} ({{point.name_p}}) </span>
<span>{point.info}</span></li>
    )}}

</div>

{selectedPoint && (
    <div className="selected-point-info">
        <h3>Информация о выбранной точке:</h3>
        <p>Название: {selectedPoint.name_p}</p>
        <p>Номер: {selectedPoint.number}</p>
        <p>Расположение: {selectedPoint.location}</p>
        <p>{selectedPoint.recom}</p>
    </div>
)}

<h3 style={{padding:20}}>Информация меридиана:</h3>
    {infoSplitList.map(item =><p>{item}</p>)}
    <h3 style={{padding:10}}>Симптомы меридиана:</h3>
    <p>{site.info} {site.info2} {site.info3}</p>
</Col>
</Row>
</Container>
) : (
    <div>
        <h4 style={{padding:10, fontFamily:'Lato'}}>
            Наиболее часто при лечении методом рефлексотерапии
используют точки, расположенные на линиях,
            которые древневосточные врачи называли меридианами.
            Они обозначали их, используя названия органов
(меридиан легких, сердца, желудка и
            т. д.).
            Также меридиан обозначают римской цифрой или одной
или двумя буквами английского,
            французского или немецкого алфавита.</h4>
        <h5 style={{padding:10, fontFamily:'Lato', textAlign:
'center'}}> Взаимоотношение болевых ощущений и меридианов: </h5>

```

```

        <img className="displayed" src={otnoshenie}
alt="otnoshenie"/>
      </div>
    )}
  </div>
);
};
export default MeridianDescription;

```

ПРИЛОЖЕНИЕ А.35

Файл meridian.js

```

import React, { useContext, useEffect, useState } from "react";
import { Container } from "react-bootstrap";
import Row from "react-bootstrap/Row";
import Col from "react-bootstrap/Col";
import MeridianList from "../meridianList";
import MeridianDescription from '../meridianDescription';
import { Context } from "../index";
import { fetchMeridian } from "../http/siteAPI";
import { observer } from "mobx-react-lite";
const Meridian = observer(() => {
  const {site} = useContext(Context)
  const [diseaseId, setDiseaseId] = useState(null)
  const [meridianLoaded, setMeridiansLoaded] = useState(false);
  useEffect(()=>{
    fetchMeridian().then(data =>
{site.setMeridians(data); setMeridiansLoaded(true)} )
  }, [])
  return(
    <div style={{padding:60,}}>
      <Container>
        <Row className="mt-2">
          <Col md='2'> {meridianLoaded && <MeridianList
onSelectMeridian={setDiseaseId}/>}</Col>

```

```

        <Col md='10'><MeridianDescription
diseaseId={diseaseId}/></Col>
    </Row>
</Container>
</div>
);
});

export default Meridian;

```

ПРИЛОЖЕНИЕ А.36

Файл main.js

```

import React from "react";
import {Container} from "react-bootstrap";

import './main.css';

const Main = () => {
    return(
        <div className = "bg">
            <Container className="mt-5">

                <div className="white" style={{padding:65}} >
                    <h1>Помощник врача-рефлексотерапевта</h1>
                </div>
                <div className="white" style={{padding:34}} >

                    <h5 style={{color: "black"}}>• </h5>
                </div>
                <div className="niz" >
                    <h4>Человеческий организм - единая</h4>
                    <h4>система, в которой всё взаимосвязанно</h4>
                    <h6 style={{color: "black"}}>>. </h6>

```

```

        <h5>Рефлексотерапия – комплекс приемов, в основе
которых</h5>
        <h5>лежит применение с лечебной целью различных
физических</h5>
        <h5>факторов воздействия на определенные точечные
участки</h5>
        <h5>поверхности тела (точки акупунктуры)</h5>
        <h6 style={{color: "black"}}>.</h6>
        <h6 style={{color: "black"}}>.</h6>
        <h6 style={{color: "black"}}>.</h6>
        <h6 style={{color: "black"}}>.</h6>
        <h6 style={{color: "black"}}>.</h6>
    </div>
    <div style={{padding:11}} >
        <h6 style={{color: "black"}}>.</h6>
    </div>
</Container>
</div>
);
};
export default Main;

```

ПРИЛОЖЕНИЕ А.37

Файл siteApi.js

```

import {$host} from "./indexA";

export const fetchMethod = async () => {
    try {
        const response = await $host.get('api/method');
        const data = response.data; // Предполагается, что данные
находятся в свойстве data объекта ответа
        return data;
    } catch (error) {
        console.error("Ошибка при получении данных:", error);
    }
}

```

```

        throw error; // Пробрасывание ошибки для обработки в
вызывающем коде
    }
}
export const fetchPoint = async () => {
    try {
        const response = await $host.get('api/point');
        const data = response.data.rows; // Предполагается, что
данные находятся в свойстве data объекта ответа
        return data;
    } catch (error) {
        console.error("Ошибка при получении данных:", error);
        throw error; // Пробрасывание ошибки для обработки в
вызывающем коде
    }
}
export const fetchMeridian = async () => {
    try {
        const response = await $host.get('api/meridian');
        const data = response.data; // Предполагается, что данные
находятся в свойстве data объекта ответа
        return data;
    } catch (error) {
        console.error("Ошибка при получении данных:", error);
        throw error; // Пробрасывание ошибки для обработки в
вызывающем коде
    }
};
export const fetchSymptom = async () => {
    try {
        const response = await $host.get('api/symptom');
        const data = response.data.rows; // Предполагается, что
данные находятся в свойстве data объекта ответа
        return data;
    } catch (error) {

```

```

        console.error("Ошибка при получении данных:", error);
        throw error; // Пробрасывание ошибки для обработки в
вызывающем коде
    }
};
export const fetchOneMeridian = async (id) => {
    try {
        const response = await $host.get('api/meridian/' + id);
        const data = response.data; // Предполагается, что данные
находятся в свойстве data объекта ответа
        return data;
    } catch (error) {
        console.error("Ошибка при получении данных:", error);
        throw error; // Пробрасывание ошибки для обработки в
вызывающем коде
    }
};

```

ПРИЛОЖЕНИЕ А.38

Файл indexA.js

```

import axios from "axios";
const $host = axios.create({
    baseURL: process.env.REACT_APP_API_URL
})
export {
    $host
}

```

ПРИЛОЖЕНИЕ Б

(обязательное)

Разработка базы знаний для интеллектуальной системы поддержки врач-
рефлексотерапевта

Презентация

**Выпускная квалификационная работа:
“Разработка базы знаний для
интеллектуальной системы поддержки
врача-рефлексотерапевта”**

Выполнил: студент гр.22ВВИм2
Касимовская А.Д.
Руководитель: Бождай А.С.

1

Цель проекта

Повышение эффективности работы врача -рефлексотерапевта за счет разработки базы знаний предметной области с последующей ее использованием в структуре интеллектуальной системы.

2

Актуальность темы

В последнее время, набирают популярность интеллектуальные и экспертные системы в области медицины различных направлений. Особую роль при этом имеет проектирование базы знаний в полной мере отображающей специфику предметной области. Востребованность систем с структурированной формализацией процессов принятия решений растет при изучении новых областей. Так, например, китайская медицина набирает популярность в развитых странах, многие люди интересуются альтернативными методами лечения и подходами к заботе о своем здоровье. Но на данный момент за пределами КНР, из-за раннего предпочтения медикаментозному лечению, существует очень маленькое количество аналогов в данной сфере и многие из них ограничены в своём доступе. Поэтому в данной нише будет востребована разработка и усовершенствование различных информационных систем.

3

Целевая аудитория

Приложение, предоставляющее информацию, может быть полезным как для профессионала в сфере китайской медицины, так и для любого человека в качестве обучающего источника. Программный продукт сможет быть инструментом для специалиста, который поможет обеспечить более качественное обслуживание пациентов, ускоряя выбор оптимального лечения.

4

Обзор предметной области

Рефлексотерапия и её применение в медицине.

Рефлексотерапия — это метод диагностики, лечения и профилактики заболеваний через стимуляцию биологически активных точек на теле.

Традиционная китайская медицина основывается на теории цельности организма и жизненной энергии, протекающей через меридианы.

Каждая такая точка связана с определенным органом или системой организма, что позволяет рефлексотерапевтам восстанавливать энергетический баланс в меридианах и улучшать функции органов.

Основой рефлексотерапии является теория традиционной китайской медицины, согласно которой здоровье определяется свободным и гармоничным течением жизненной энергии через меридианы.

Нарушение этого потока вызывает заболевания.

Рефлексотерапевты восстанавливают баланс, что способствует выздоровлению.

5

Обзор предметной области

Интеллектуальные системы в медицине.

Интеллектуальная система — это техническая или программная система, способная решать задачи на основе знаний, хранящихся в её памяти. Основные компоненты таких систем:

- База знаний: содержит правила и информацию о человеческом опыте и знаниях
- Механизм вывода решений: анализирует данные и выдает рекомендации.
- Интеллектуальный интерфейс: обеспечивает взаимодействие пользователя с системой.

Применение интеллектуальных систем в медицине значительно улучшает диагностику, лечение и управление медицинскими данными.

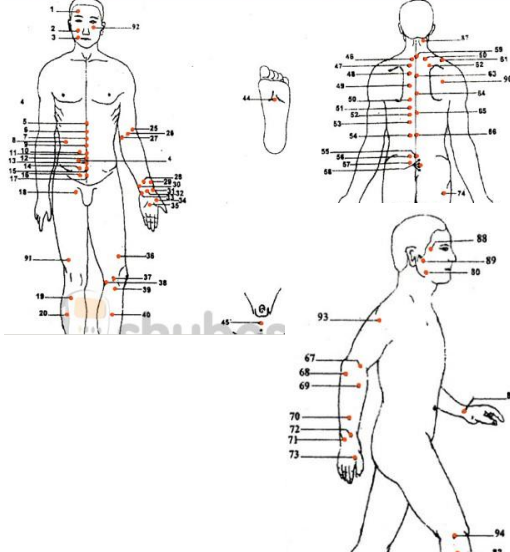
Примеры использования включают:

- 1) Диагностика заболеваний: системы поддержки принятия решений анализируют данные пациента и сопоставляют их с базами данных симптомов и историй болезней.
- 2) Управление медицинскими данными: электронные медицинские записи позволяют эффективно хранить и анализировать информацию о пациентах.
- 3) Увеличение доступа к медицинским услугам: дистанционные консультации и телемедицина делают медицинскую помощь доступнее.

Перспективы развития включают улучшение алгоритмов машинного обучения, интеграцию новых данных и расширение использования телемедицины.

6

Демонстрация аналогов



1. Болевая точка
 2. Ян Цзюэ - VB.14
 3. Цинь Ляо - IG.18
 4. Да Цинь - E.4
 5. Цинь Чжун - PC.41
 6. Шань Бэй - VC.13
 7. Чжун Бэй - VC.12
 8. Сяо Бэй - VC.10
 9. Чжун Мань - F.13
 10. Шунь Фань - VC.9
 11. Тян Шу - E.25
 12. Шунь Цюэ - VC.8
 13. Инь Цзюэ - VC.7
 14. Цяо Хай - VC.6
 15. Шунь Дяо - E.28
 16. Гоуань Цзюань - VC.4

1. Чжун Цзюэ - P.5
 2. Цяо Цзюэ - MC.3
 3. Сяо Хай - IG.8
 4. Нэй Гуань - MC.6
 5. Ян Цзюэ - P.7
 6. Цунь Цзюэ - C.5
 7. Ян Цзюэ - P.9
 8. Да Линь - MC.7
 9. Шунь Мань - C.7
 10. Ян Цзюэ - P.10
 11. Сяо Гуань - MC.8
 12. Сяо Хай - RP.10
 13. Цяо Цзюэ - F.8
 14. Инь Гуань - RP.9
 15. Инь Линь Цзюэ - RP.9
 16. Да Линь - RP.8

1. Гань Шу - VB.18
 2. Дянь Шу - VB.19
 3. Ян Шу - V.20
 4. Ян Шу - V.21
 5. Шунь Шу - V.23
 6. Ян Шу Гуань - VC.3
 7. Да Чан Шу - V.25
 8. Пан Гуан Шу - V.28
 9. Шань Ляо - V.31
 10. Да Чжун - VC.14
 11. Цзюэ Бай Шу - IG.14
 12. Цзюэ Бай Шу - VB.21
 13. Шу Фань - V.41
 14. Цзюэ Шу - VC.12
 15. Чжун Шу - VC.9
 16. Цзюэ Шу - VC.6
 17. Мань Мань - VC.4
 18. Шунь Сяо Шу - IG.10
 19. Шунь Шу - C.3
 20. Цяо Чжун - GI.11
 21. Бай Гуань - TR.5
 22. Ян Ляо - VC.6
 23. Ян Чжун - TR.4
 24. Хэ Гу - GI.4
 25. Хуань Шу - VB.30

1. Чжун Шу - V.36
 2. Ян Шу - V.37
 3. Ян Шу - V.40
 4. Чжун Шу - V.57
 5. Шунь Шу - V.62
 6. Цзюэ Шу - E.6
 7. Бай Гуань - P.9
 8. Ян Линь Цзюэ - VB.34
 9. Ян Цзюэ - VB.35
 10. Сяо Чжун - VB.39
 11. Кунь Шу - V.60
 12. Цзюэ Шу - VB.40
 13. Шу Шу - VB.20
 14. Бай Шу - E.1
 15. Цзюэ Чжун Шу - FN.25
 16. Тян Цзюэ - IG.11
 17. Лян Цзюэ - E.34
 18. Сяо Шу - E.2
 19. Ян Шу - FN.73
 20. Сяо Ян Шу - VB.33

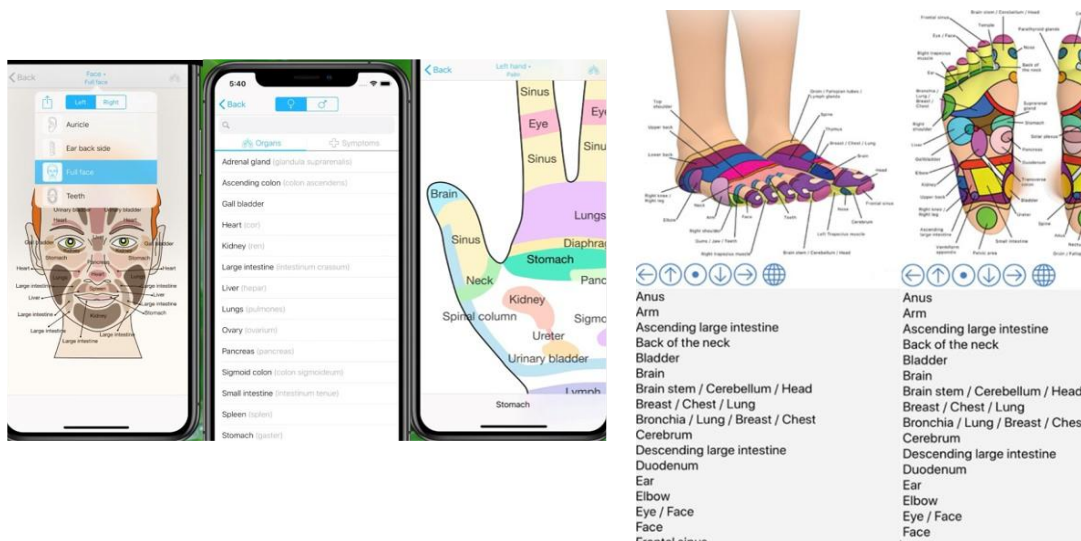
ГИПЕРТОНИЯ
 Сочетания точек:
 а. Юн Цзюэ + Юн Цзюэ (44+44)
 б. Нэй Гуань + Цяо Чжун (28+69) (с обеих сторон)
 в. Цзюэ Сан Ли + Сяо Инь Цзюэ (19+41)
 г. Сяо Инь Цзюэ + Юн Цзюэ (41+44)
 Метод: каждую пару по 20 мин. на 1 или 2 режиме, 1 или 2 раза в день, сила воздействия умеренная.

СКЛЕРОЗ МОЗГОВЫХ АРТЕРИЙ
 Сочетания точек:
 а. Фань Чжун + Цзюэ Цзюэ (87+61)
 б. Сяо Чжун + Сяо Инь Цзюэ (84+41)
 в. Шунь Шу + Шунь Шу (54+54)
 Метод: каждую пару по 10 мин. на 2,4-ом режиме, 1 или 2 раза в день, сила воздействия умеренная.

ОСЛОЖНЕНИЕ ПОСЛЕ ИНСУЛЬТА
 Сочетания точек:
 а. Гань Шу + Гань Шу (50+50)
 б. Пу Шу + Пу Шу (52+52)

7

Демонстрация аналогов



Organs
 Adrenal gland (glandula suprarenalis)
 Ascending colon (colon ascendens)
 Gall bladder
 Heart (cor)
 Kidney (ren)
 Large intestine (intestinum crassum)
 Liver (hepar)
 Lungs (pulmones)
 Ovary (ovarium)
 Pancreas (pancreas)
 Sigmoid colon (colon sigmoideum)
 Small intestine (intestinum tenue)
 Spleen (splen)
 Stomach (gaster)

Organs
 Sinus
 Eye
 Sinus
 Sinus
 Lungs
 Diaphragm
 Stomach
 Panc
 Neck
 Kidney
 Ureter
 Urinary bladder
 Stomach
 Lymph

Organs
 Anus
 Arm
 Ascending large intestine
 Back of the neck
 Bladder
 Brain
 Brain stem / Cerebellum / Head
 Breast / Chest / Lung
 Bronchia / Lung / Breast / Chest
 Cerebrum
 Descending large intestine
 Duodenum
 Ear
 Elbow
 Eye / Face
 Face
 Frontal sinus

Organs
 Anus
 Arm
 Ascending large intestine
 Back of the neck
 Bladder
 Brain
 Brain stem / Cerebellum / Head
 Breast / Chest / Lung
 Bronchia / Lung / Breast / Chest
 Cerebrum
 Descending large intestine
 Duodenum
 Ear
 Elbow
 Eye / Face
 Face
 Frontal sinus

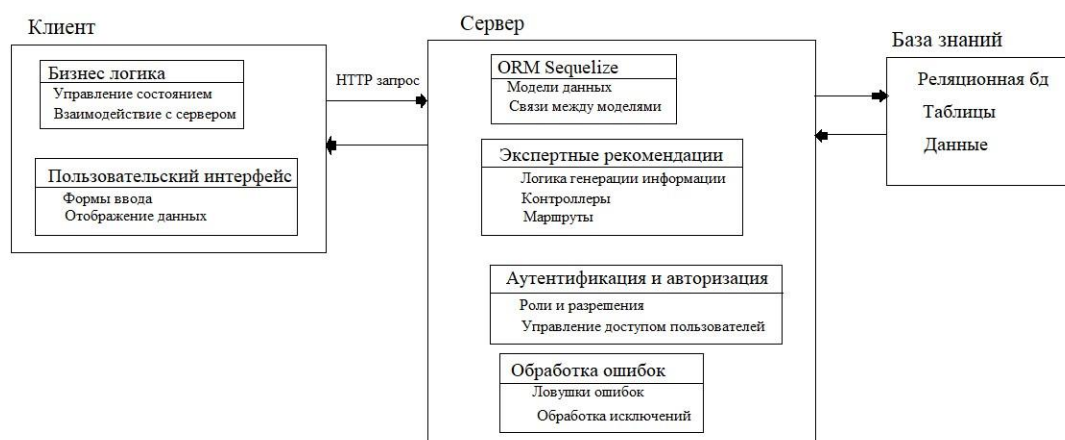
8

Обзор аналогов

Критерии	Информационная система «Shuboshi»	Приложение «Massage Maps»	Приложение «Foot Reflexology Chart»
Поисковая строка	-	+	-
Информация о меридианах	-	-	-
Описание метода воздействия	+	-	-
Описание точек	-	-	-
Доступны все области тела	+	-	-
Графическое отображение точек	+	+	+

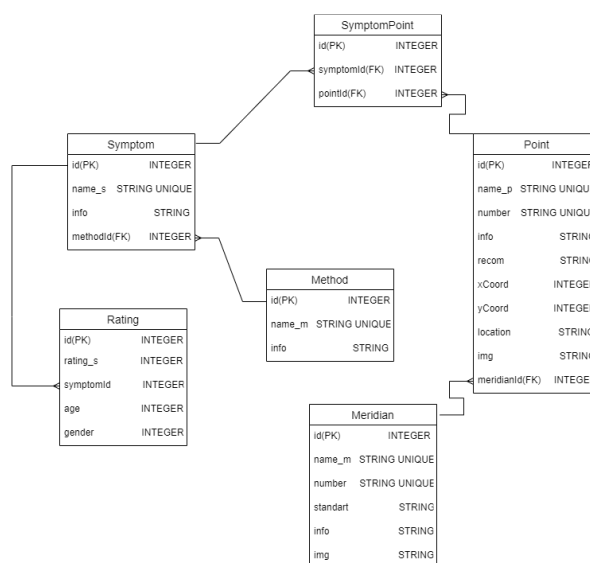
9

Структура системы



10

Модель данных



11

Перечень функций

Разрабатываемое программное обеспечение должно представлять собой web-приложение, которое включает в себя базу знаний, основанную на связанных данных для врача-рефлексотерапевта.

Программа должна обеспечивать выполнение следующих функций:

- отображение всех данных пользователю в удобном структурированном виде,
- принятие запроса пользователя к базе данных,
- отображение результата.

При работе с сайтом пользователь получает обратную связь от своих действий. Это реализуется следующими способами: внешний вид элементов интерфейса изменяется при нажатии на них; при отправке запроса выводится результат.

Фильтрация данных. Пользователь вводит поисковый запрос, и данные фильтруются на основе этого запроса.

12

Модель машинного обучения в системе

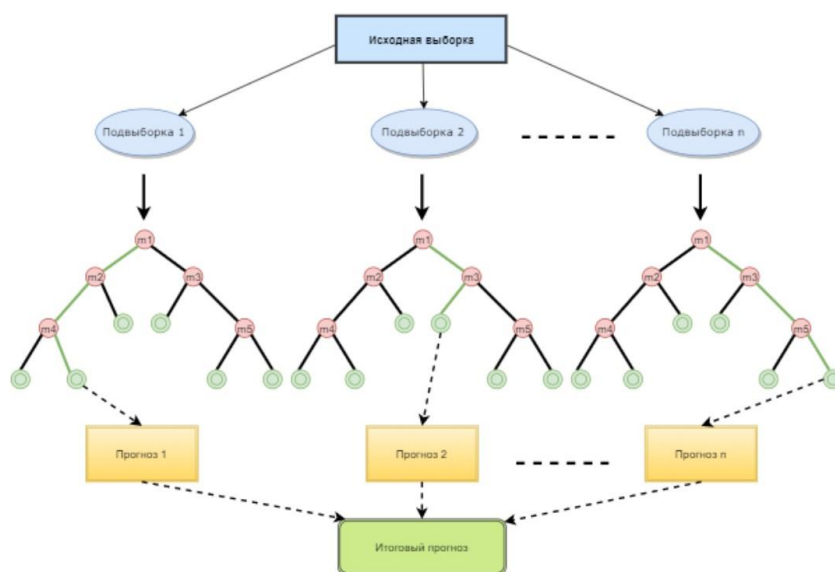
Для предсказания результата лечения для пациента при определенном симптоме в системе анализа используется алгоритм «случайный лес», являющийся методом машинного обучения.

Цель построение такого алгоритма – это создание модели, которая предсказывает значения целевой переменной в зависимости о нескольких переменных на входе. Обучающими данными для модели являются данные пройденных приемов лечения.

В исходной выборке используются данные об объектах с известными значениями признака (symptom, age, gender) и целевой переменной (rating).

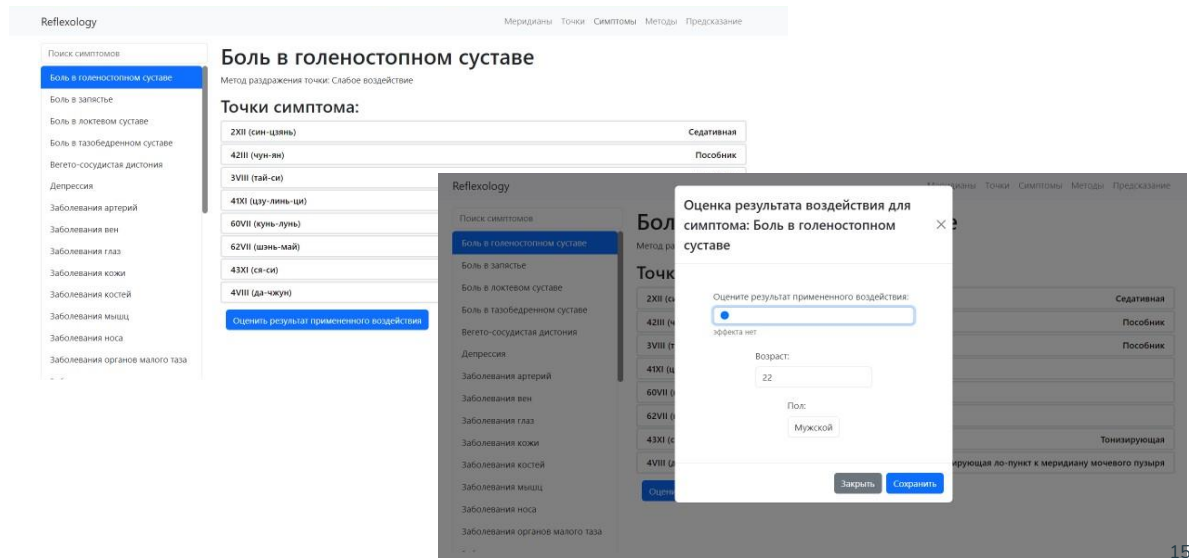
13

Модель машинного обучения в системе

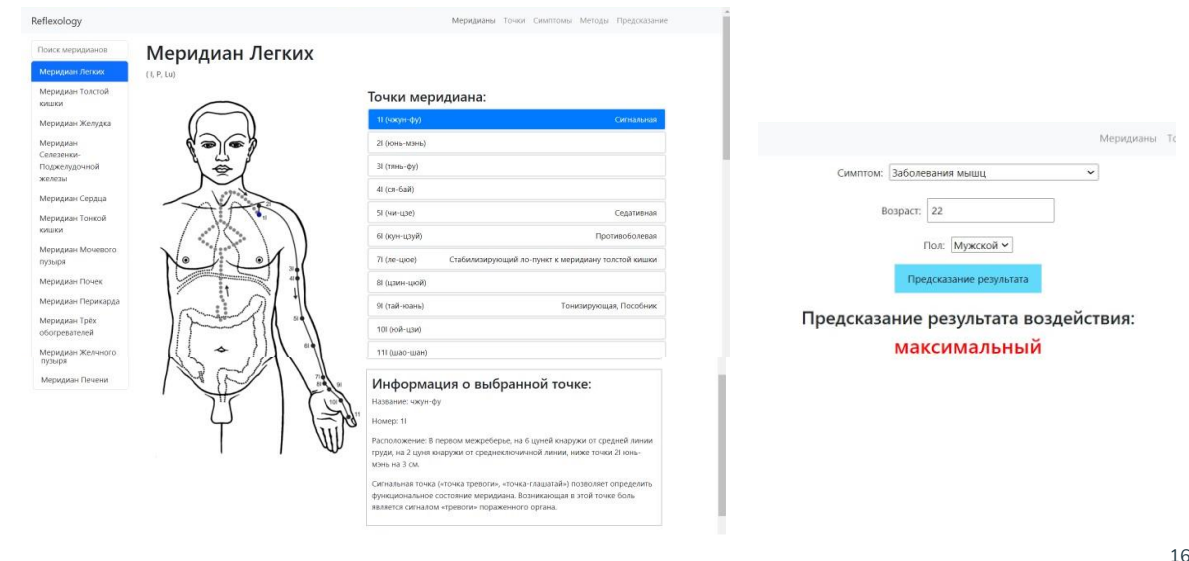


14

Интерфейс приложения



Интерфейс приложения



Заключение

Исследовано развитие интеллектуальных систем и рассмотрен принцип работы врача-рефлексотерапевта для удобной формализации данных.

Предложен собственный вариант базы знаний для интеллектуальной системы поддержки врача-рефлексотерапевта.

Проанализированы существующие решения и выявлены их недостатки для улучшения разработки.

Основные достоинства разработанной базы знаний включают доступность, понятную логику использования и полезные справочные функции.

Для демонстрации возможностей создано программное обеспечение с управляющей базой знаний, с организационной системой запросов и визуализацией контента.

На данный момент подсистема экспертных рекомендаций реализована в виде механизма отдельных поисковых запросов. В перспективе планируется объединить их в единую подсистему с более продвинутым интерфейсом формирования вопросов и улучшенным машинным обучением, учитывающим смежные заболевания, комбинации точек, сезон года и время суток, когда проводилось лечение.

17

Спасибо за внимание!

18