

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

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

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки

09.03.02
«Информационные
системы и технологии»

БАКАЛАВРСКАЯ РАБОТА

на тему

**Web-портал для поиска и заказа медицинских
услуг**

Студент

(подпись, дата) Пустобаева Анастасия Александровна
(ФИО полностью)

Руководитель

(подпись, дата) Шибанов С. В.
(фамилия, инициалы)

Нормоконтролёр

(подпись, дата) Такташкин Д.В.
(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой

(подпись) Макарычев П.П.
(фамилия, инициалы)

Работа защищена с отметкой

(протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

(подпись) Попова Н.А.
(фамилия, инициалы)

Пенза, 2021

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
П.П. Макарычев
«__» _____ 2021 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Пустобаева Анастасия Александровна гр. 17ВИ1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы «Web-портал для поиска и заказа медицинских услуг»

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

2. Руководитель работы к.т.н., доцент кафедры «МО и ПЭВМ» Шибанов С. В.

3. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение разработки: web-портал для поиска и заказа медицинских услуг
предназначен для автоматизации записи на прием и поиска необходимой информации для
пользователя.

Средства выполнения: Операционная система Windows (версия 10)

Средства разработки: WebStorm 2021.1.2, MySQL, Framework React

Язык программирования: JavaScript

Технологии разработки: ООП, UML, MVC

4. Перечень подлежащих разработке вопросов _____

Анализ предметной области и постановка задачи на разработку

Анализ требований на разработку

Разработка информационного и программного обеспечения

Контроль качества программного обеспечения

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и постановка задачи на разработку	24	14.04.21 - 19.04.21	
2	Анализ требований на разработку	31	19.04.21 - 01.05.21	
3	Разработка информационного и программного обеспечения	15	01.05.21 - 01.06.21	
4	Контроль качества программного обеспечения	15	01.06.21 - 14.06.21	

Дата выдачи задания "28" декабря 2020

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____
(подпись, дата)

Реферат

Пояснительная записка содержит 127 листов, 66 таблиц, 36 рисунков, 21 использованный источник и 6 приложений.

АВТОМАТИЗАЦИЯ, БД, СУБД, SQL, MVC, UML, JS, JAVASCRIPT.

Объект исследования: web-портал для поиска и заказа медицинских услуг.

Цель работы: реализовать web-портал для поиска и заказа медицинских услуг.

Технологии разработки: ООП, MVC, JAVASCRIPT, WebStorm 2021.1.2, Framework React.

Задание: спроектировать и реализовать интерфейс web-портала для пользователей, логику подключения к программным средствам систем управления базами данных.

Результаты работы: спроектирован и разработан web-портал для поиска и заказа медицинских услуг.

					ПГУ 09.03.02–8БР171.115 ПЗ			
Изм.	Лист	№ докум.	Подпис	Дата	«Web-портал для поиска и заказа медицинских услуг» Пояснительная записка	Лит.	Лист	Листов
Разраб.		Пустобаева А.А.						
Провер.		Шибанов С.В.					4	127
						Группа 17ВИ1		
Н. Контр.		Такташкин Д.В.						
Утверд.		Макарычев П.П.						

Содержание

Введение	7
1 Анализ предметной области и постановка задачи на разработку	9
1.1 Анализ предметной области	9
1.1.1 Анализ структуры предметной области	9
1.1.2 Анализ бизнес-процессов предметной области	9
1.1.3 Анализ регламентирующих документов	12
1.2 Сравнительный анализ аналогичных разработок	15
1.2.1 Выбор средств систем-аналогов	15
1.2.2 Обзор основных возможностей систем-аналогов	16
1.2.3 Результаты сравнительного анализа систем-аналогов	22
1.3 Постановка задачи на разработку	26
1.3.1 Цели разработки и назначение	26
1.3.2 Классы пользователей	26
1.3.3 Перечень автоматизируемых функций	27
1.4 Планирование и функционально-стоимостный анализ процесса разработки	28
1.4.1 Основы функционально-стоимостного анализа	28
1.4.2 Планирование процесса разработки	29
1.4.3 Анализ временных и финансовых затрат	30
2 Анализ требований на разработку	32
2.1 Функциональные требования	32
2.2 Построение концептуальной модели	38
2.3 Требования к обобщенной архитектуре приложения	39
2.4 Обоснование выбора технологий и средств разработки	42
2.4.1 Обоснование выбора системы управления базами данных	42
2.4.2 Обоснование выбора языковых и инструментальных средств	43
2.4.3 Обоснование выбора фреймворка для разработки приложения	44
2.5 Требования к аппаратно-программному обеспечению	46
2.5.1 Требования к аппаратно-техническому обеспечению	46
2.5.2 Требования к системному программному обеспечению	46
3 Разработка информационного и программного обеспечения web-портала	48
3.1 Разработка архитектуры web-портала	48
3.2 Разработка базы данных web-портала	50
3.3 Разработка программного обеспечения серверного приложения web-портала	51
3.4 Разработка пользовательского интерфейса web-портала	54
4 Контроль качества программного обеспечения	57
4.1 Планирование процесса контроля качества программного обеспечения	57
4.1.1 Мероприятия по контролю качества программного обеспечения	58

4.2 Тестирование программного обеспечения	60
Список использованных источников	71
Приложение А. Описание диаграмм деятельности	74
Приложение Б. Описание концептуальной модели	84
Приложение В. Описание сущностей базы данных	90
Приложение Г. Код создания базы данных	96
Приложение Д. Код создания сервера	100
Приложение Е. Код создания клиентской части	108

Введение

В современном мире техника начала играть важную роль в жизни человека. Люди используют ее не только для игр и развлечений, но и для упрощения жизни. Некоторые отрасли, такие как машиностроение, медицина, промышленное производство и так далее, стараются автоматизировать производственные процессы для того, чтобы:

- сокращать время производства продукции или на получение услуги;
- выполнять работу, которая невыполнима для человека;
- вести различные расчетные операции.

Главной целью выпускной квалификационной работы является автоматизация записи на получение медицинских услуг. Актуальность выбранной темы связана с тем, что все люди болеют и им требуется помощь специалиста, а для того чтобы получить ее вовремя и не тратить время на стояние в очередях и создается электронная запись, которая упрощает во многом жизнь человеку.

Для достижения поставленной цели необходимо решить следующие задачи:

- выполнить анализ предметной области;
- выполнить постановку задачи на разработку;
- провести сравнительный анализ систем-аналогов;
- выполнить анализ требований на разработку;
- разработать информационное и программное обеспечение;
- выполнить контроль качества программного обеспечения;
- провести тестирование программного обеспечения.

В результате выполнения выпускной квалификационной работы были реализованы следующие функции:

- регистрация и вход в систему пользователей;
- запись на получение медицинской услуги;
- просмотр медицинских учреждений города;
- просмотр списка врачей;
- просмотр медицинских услуг, а также их стоимости;

- возможна регистрация медицинских учреждений;
- возможен просмотр аналитики.

В дальнейшем планируется разработка следующих функций:

- подбор ближайшего медицинского учреждения;
- отображение медицинских организаций на карте;
- подбор специалиста по симптомам;
- онлайн консультация;
- возможность записей в регионах.

Функционал может быть расширен при желании разработчика.

1 Анализ предметной области и постановка задачи на разработку

1.1 Анализ предметной области

1.1.1 Анализ структуры предметной области

Портал – сайт в интернете, который предназначен для того, чтобы пользователь смог получить желаемые им услуги, которые предоставляются на данном интернет-сервисе.

Портал медицинских услуг предназначен как для людей, которые хотят записаться на прием к врачу или на лабораторное обследование, так и для администратора учреждения.

В первом случае данный портал будет решать проблему очередей в больницах. Помимо этого, люди, которые приходят в больницу уже больными не смогут заразить других, стоя в очереди, ведь большинство болезней передаются воздушно-капельным путем. Таким образом, человек, который пришел обследоваться не будет контактировать с заболевшими людьми и заболеваемость будет ниже. Каждый пользователь будет приходить в больницу только в определенное время.

Во втором случае администратор сможет в удобном виде анализировать данные приема и изменять устаревшую информацию о медицинском учреждении.

В третьем случае, можно будет получить платную и бесплатную онлайн консультацию многих врачей, в зависимости от их специальностей.

1.1.2 Анализ бизнес-процессов предметной области

Бизнес-процесс представляет собой совокупность взаимосвязанных между собой работ, которые направлены на создание определенного продукта. Они предназначены для достижения какого-либо желаемого результата [1].

Для анализа бизнес-процессов были выделены классы действующих лиц предметной области, которые представлены в таблице 1.

Таблица 1 – Классы действующих лиц предметной области

№ п/п	Категория пользователей	Информационные задачи
1	Медработник	Может просматривать записи к себе на приём, свое расписание, консультировать пользователей онлайн.
2	Системный администратор	Предоставление доступа к загрузке/выгрузке информации для представителя медицинского учреждения, анализ данных
3	Пользователь (пациент)	Может записываться на медицинскую услугу, просматривать расписание врачей, информацию о медицинских учреждениях, записываться на онлайн консультации.
4	Представитель мед. организации	Загрузка/выгрузка информации о медицинском учреждении и поддержание актуальных данных

Бизнес-процессы системы представлены в таблице 2.

Таблица 2 – Бизнес-процессы системы

№ п/п	Бизнес-процесс	Исполнитель	Поставщик	Потребитель
1	Добавить медицинское учреждение в систему	Системный администратор	Медработник	Пользователь
2	Добавить расписание предоставляемых услуг в систему	Системный администратор	Медработник	Пользователь
3	Добавить медицинские услуги и их стоимость в систему	Системный администратор	Медработник	Пользователь
4	Изменение данных о медицинском учреждении	Системный администратор	Медработник	Пользователь

Продолжение таблицы 2

№ п/п	Бизнес-процесс	Исполнитель	Поставщик	Потребитель
5	Изменение данных о медицинских работниках	Системный администратор	Медработник	Пользователь
6	Регистрация в системе	Пользователь	Системный администратор	Пользователь
7	Запись на лабораторное обследование	Пользователь	Системный администратор	Пользователь
8	Запись на прием к врачу	Пользователь	Системный администратор	Пользователь
9	Запись на онлайн-консультацию	Пользователь	Системный администратор	Пользователь
10	Анализ внутри системы по каждому медицинскому учреждению (загруженности сотрудников и востребованность услуг)	Системный администратор	Медработник	Пользователь
11	Изменение данных о мед. услугах	Системный администратор	Медработник	-
12	Импорт данных о мед. учреждении	Системный администратор	Медработник	-
13	Экспорт данных о мед. учреждении	Системный администратор	Медработник	-

На основе выделенных бизнес-процессов в таблице 1.2, на этапе анализа требований определены функциональные возможности web-портала.

1. 1. 3 Анализ регламентирующих документов

Регламентирующие документы необходимы для того, чтобы регулировать деятельность организации в различных сферах. Документы являются обязательными, официальными и многократно применяемыми.

В ходе анализа были выявлены такие регламентирующие документы, как «Нормативно-правовые документы медицинской организации» и «Требования к официальному сайту медицинской организации». Далее они будут рассмотрены подробнее.

К нормативно-правовым документам относится Федеральный закон №323 «Об основах охраны здоровья граждан». При оказании медицинской помощи учитываются следующие документы, которые регламентируют деятельность медицинской организации:

- закон об обязательном медицинском страховании;
- закон о защите прав потребителей;
- постановление о программе государственных гарантий оказания гражданам бесплатной медпомощи на соответствующий год;
- правила предоставления клиниками медицинских услуг на платной основе;
- порядок и стандарты медицинской помощи;
- санитарные требования к медицинским учреждениям;
- политика информационной безопасности и др. [2].

Каждое медицинское учреждение должно обеспечить сбор, учет и хранение документации, однако единого порядка хранения пока нет, существующий Приказ Минздрава №1030 носит рекомендательный характер. Но при организации работы с медицинскими документами необходимо учитывать:

- Закон №323, где указаны требования к обязательности учета, хранения документов и оформления бланков строгой отчетности, при хранении учетной документации также следует учитывать правила ее выдачи пациенту или его представителю, основания и порядок выдачи документов (их копий или

выписки из них) устанавливаются уполномоченным органом исполнительной власти;

- ФЗ №184, который касается регулирования электронных систем документооборота в техническом плане.

Кроме того, медицинское учреждение должно обеспечить сохранность архивных документов в течение установленного законом срока, общие требования определены ФЗ № 125 «Об архивном деле», а также отдельными нормативными актами [3].

Помимо этих документов существуют также требования к официальным сайтам, которые отражены в Приказе Министерства здравоохранения РФ от 30 декабря 2014 г. № 956н. Согласно закону от 21 ноября 2011 г. № 323-ФЗ любая медицинская организация должна предоставить открытую информацию о своей деятельности, в том числе и через сайт.

Информация делится на такие подгруппы, как:

- о медицинской организации;
- для пациентов;
- для специалистов;
- для медицинских работников;
- лекарственное обеспечение;
- вышестоящие и контролирующие органы;
- отзывы.

Далее будет рассмотрена каждая подгруппа отдельно.

Информация о медицинской организации должна содержать:

- полное и краткое название медицинской организации, а также ее структурных подразделений (при наличии);
- дата государственной регистрации;
- телефоны, адреса, карта;
- режим работы;
- данные об учредителе и руководителе с контактами последнего;
- лицензию на медицинскую деятельность.

Информация для пациентов должна содержать:

1) условия оказания бесплатной медицинской помощи гражданам РФ в рамках территориальной программы государственных гарантий, а именно:

- виды, объем, порядок и условия;
- доступность и качество медпомощи;
- порядок записи на первичный прием или обследование;
- правила подготовки к диагностическим исследованиям;
- правила и сроки госпитализации;

2) диспансеризация:

– сроки, порядок и результаты диспансеризации населения, прикрепленного к МО;

3) платные услуги:

- правила и форма предоставления мед. услуг;
- перечень услуг;
- цены и порядок оплаты;
- образец договора;
- данные о медицинских работниках, оказывающих платные услуги, их квалификация и часы приема.

В информации для специалиста размещается перечень утвержденных порядков и стандартов со ссылками на соответствующие документы.

В информации о медицинских работниках указываются следующие данные: ФИО, специализация, должность, график и часы приема. Также нужно указать сведения из документа об образовании.

В информации о лекарственном обеспечении вносится перечень лекарственных препаратов, которые назначаются по решению медицинской комиссии, а также отпускаемые бесплатно или со скидкой 50 %.

В информации о вышестоящих и контролирующих органах размещаются контактные данные, а также адреса: органа исполнительной власти субъекта РФ в сфере охраны здоровья и территориальных органов Федеральной службы по надзору в сфере здравоохранения и прав потребителей.

Говоря об отзывах, необходимо отметить, чтобы была форма для их заполнения пользователями, а сами отзывы были в открытом доступе, чтобы каждый мог с ними ознакомиться [4].

1.2 Сравнительный анализ аналогичных разработок

1.2.1 Выбор средств систем-аналогов

Сравнительный анализ является одной из необходимых составных частей при создании приложений или сайтов. Необходимо выявить достоинства и недостатки систем, которые уже существуют, чтобы суметь сделать то, чего еще нет, совершенствуя свою систему и не допуская минусов у себя. Аналогов разрабатываемому portalу медицинских услуг очень много.

Перечень систем-аналогов, которые будут рассмотрены:

- К–Врачу;
- Электронная регистратура;
- Кубань–online;
- MEDODS;
- DocDoc.ru.

Дополнительная информация о системах представлена в таблице 3.

Таблица 3 – Дополнительная информация о системах

Системы	Критерии		
	Разработчик	Доступность	Платформа
Кубань–online	МИАЦ	Бесплатная, онлайн-консультации платные	Поддерживает работу через сайт, приложений нет.
К–Врачу	СВАН	Работа с порталом бесплатная, запись на прием есть платная и бесплатная	Есть приложения для IOS и Android, также поддерживает работу через сайт на компьютере.

Продолжение таблицы 3

Системы	Критерии		
	Разработчик	Доступность	Платформа
Электронная регистратура	«Мастер Лаб» (MASTER LAB)	Работа с порталом бесплатная, запись на прием платная, онлайн-консультации платные	Поддерживает работу через сайт, приложений нет.
MEDODS	ООО МЕДОДС	3900 в месяц (возможна бесплатная пробная версия)	Есть приложения для IOS и Android, также поддерживает работу через сайт на компьютере.
DocDoc.ru	SBERHEALTH.RU	Бесплатная, онлайн-консультации платные	Есть приложения для IOS и Android, также поддерживает работу через сайт на компьютере.

В ходе сравнительного анализа были подробно изучены основные возможности каждой из систем-аналогов.

1.2.2 Обзор основных возможностей систем-аналогов

Ниже будут представлены интерфейсы и функции систем, которые были взяты для сравнения.

Интерфейс системы «К–Врачу» представлен на рисунке 1.

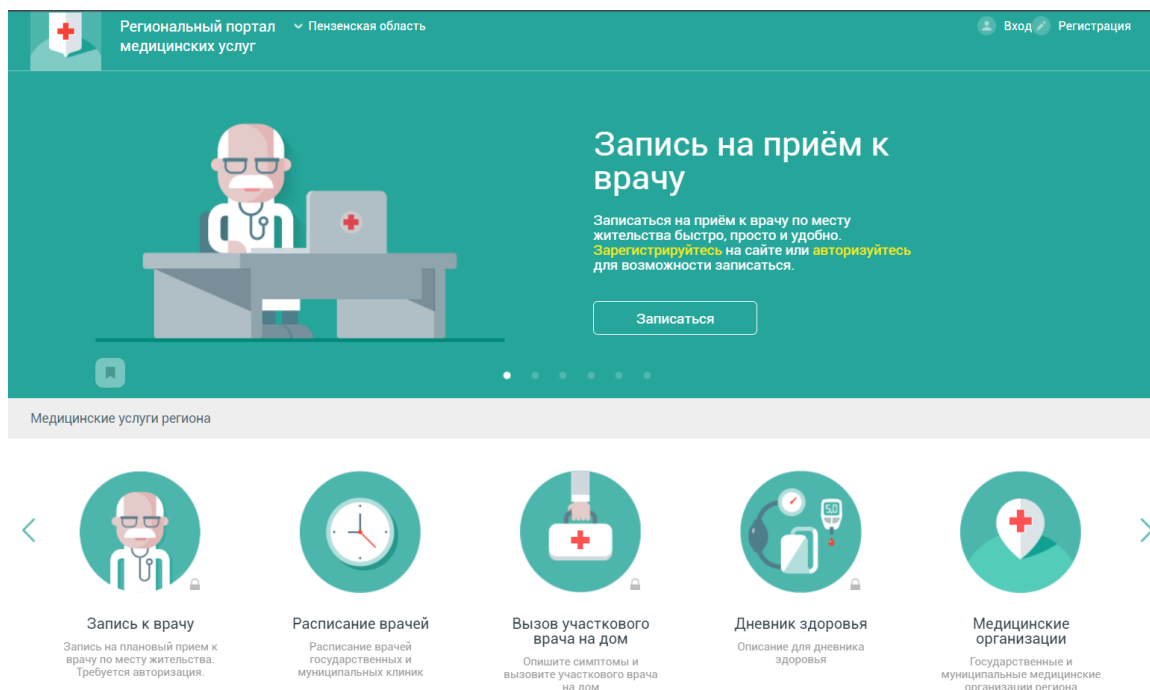


Рисунок 1 – Интерфейс системы «К–Врачу»

Система «К–Врачу» позволяет незарегистрированным пользователям просматривать:

- медицинские организации региона;
- участок прикрепления;
- расписание врачей;
- запись на вакцинацию.

Для зарегистрированных пользователей, помимо вышеперечисленных, доступны следующие функции:

- запись к врачу;
- вызов участкового врача на дом;
- дневник здоровья.

Описанная выше система обладает базовым набором функций, которые присущи каждому сайту.

Интерфейс системы «Электронная регистратура» представлен на рисунке 2.

Шаг 1 Шаг 2 Шаг 3 Шаг 4 Шаг 5 Шаг 6

Муниципальное образование Медицинское учреждение Врач Время Пациент Запись

Укажите населенный пункт
Выберите регион и населенный пункт, в котором предполагается посетить медицинское учреждение

Башмаковский район	Камешкирский район	Никольский район
Бековский район	Колышлейский район	Пачелмский район
Белинский район	Кузнецкий район	Пенза
Бессоновский район	Лопатинский район	Пензенский район
Вадинский район	Лунинский район	Сердобский район
Городищенский район	Малосердобинский район	Сосновоборский район
Заречный	Мокшанский район	Спасский район
Земетчинский район	Наровчатский район	Тамалинский район
Иссинский район	Неверкинский район	Шемышейский район

Рисунок 2 – Интерфейс системы «Электронная регистратура»

«Электронная регистратура» имеет такую вкладку, как «Услуги портала», где предоставлены услуги для клиник, врачей и пациентов. Для клиник есть такие услуги, как:

- добавить свою клинику на портал;
- телемедицинский сервис для клиники;
- мобильное приложение для клиники;
- бесплатный сайт для клиники;
- продвижение медицинских медучреждений и их услуг в интернете;
- интеграция с медицинской системой;
- виджет записи на прием.

Для врачей предусмотрены следующие услуги:

- бонусные программы портала;
- мобильное приложение для врачей.

Для пациентов существуют такие услуги, как:

- бесплатная медицинская консультация на портале;
- онлайн и видео консультация;
- мобильное приложение для пациентов;

– ОТЗЫВЫ О КЛИНИКАХ И ВРАЧАХ.

«Услуги портала» представлены на рисунке 3.

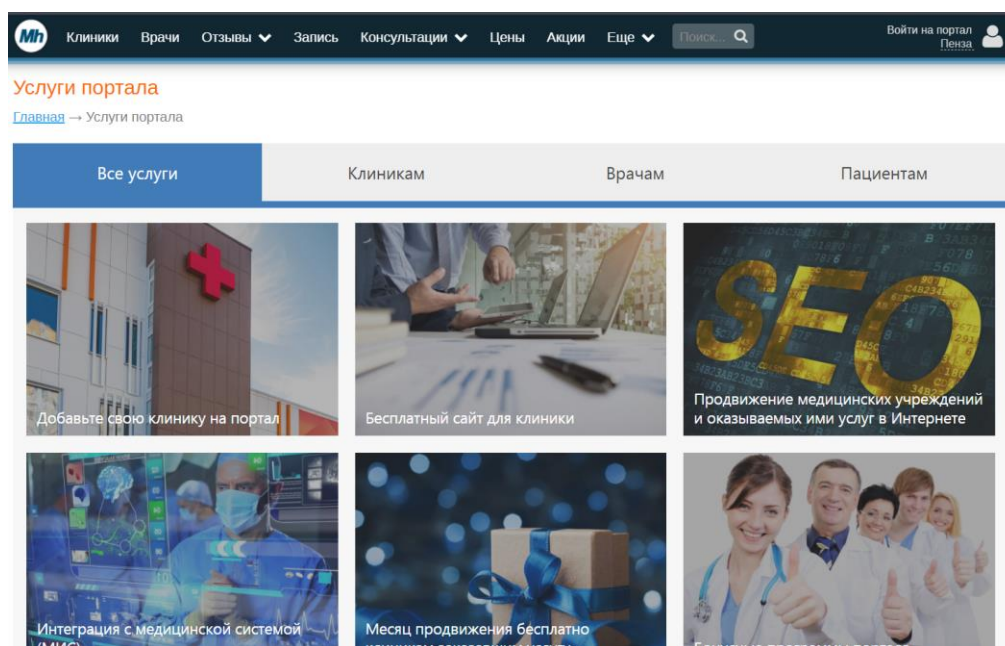


Рисунок 3 – Услуги портала «Электронная регистратура»

Интерфейс системы «Кубань–online» представлен на рисунке 4.

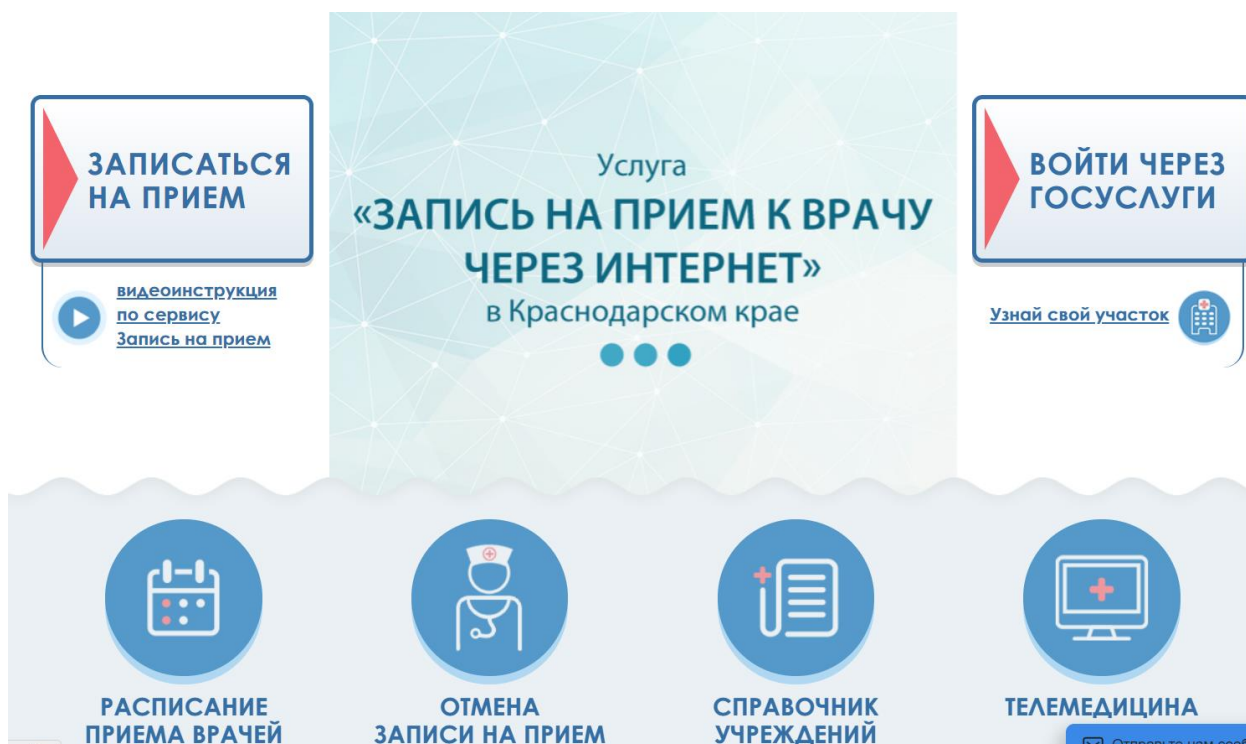


Рисунок 4 – Интерфейс системы «Кубань-online»

Отдельно хотелось бы указать, что сайт системы «Кубань–online» имеет версию для слабовидящих, что позволяет людям, у которых проблемы со

зрением самостоятельно и с удобством пользоваться услугами, которые предоставляет данная система.

Помимо этого, у системы на главной странице предусмотрены следующие возможности:

- вопросы и ответы;
- новости портала;
- новости здравоохранения;
- опрос.

Наибольший интерес вызывает опрос, который предназначен для пользователей данной системы. Он поможет получать обратную связь об удобстве пользования порталом, что является не менее важным пунктом, как для пациентов, так и для врачей. Опрос представлен на рисунке 5.

ОПРОС

С какими трудностями Вы столкнулись при использовании сервиса «Запись на прием к врачу в электронном виде»?

- ☐ Отказ в приеме
- ☐ Отсутствие актуальных расписаний медучреждений
- ☐ Проблемы записи на прием к врачу через инфомат
- ☐ Проблемы в работе портала
- ☐ С трудностями не столкнулся

Проголосовать

Ваше мнение важно для нас.

Рисунок 5 – Опрос системы «Кубань-online»

Интерфейс системы MEDODS представлен на рисунке 6.

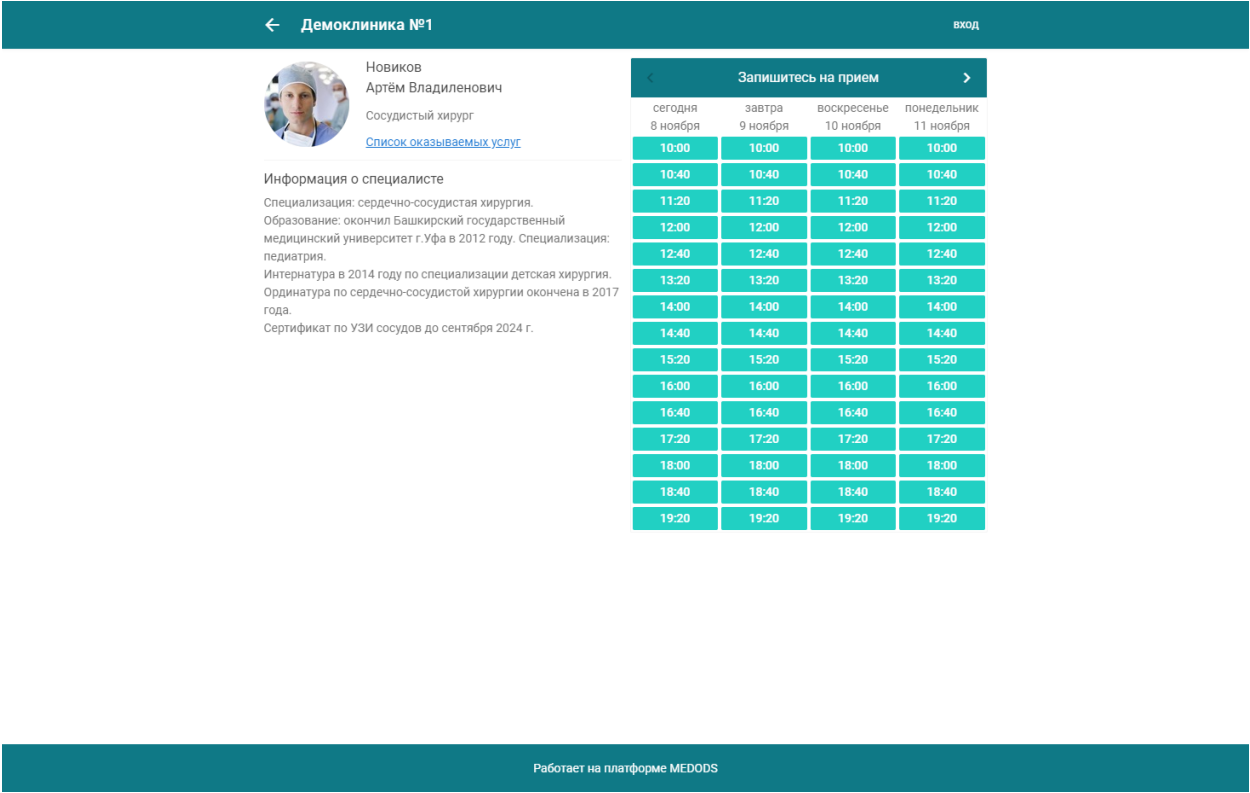


Рисунок 6 – Опрос системы «MEDODS»

Интерфейс системы DocDoc.ru представлен на рисунке 7.

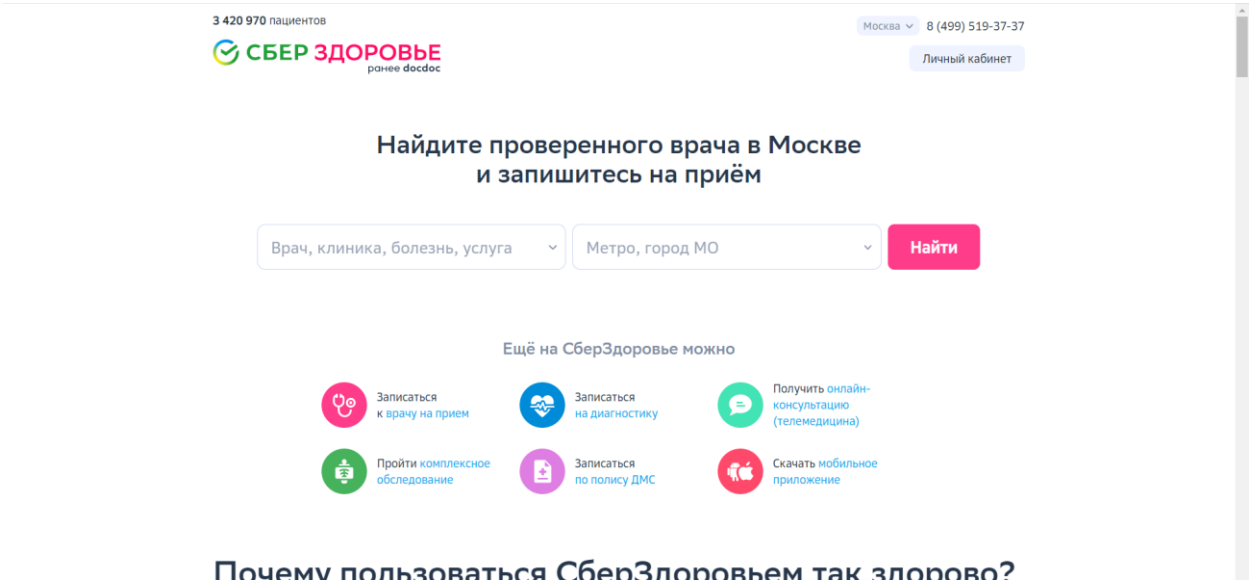


Рисунок 7 – Интерфейс системы «DocDoc.ru»

На главной странице системы DocDoc.ru есть статистика, которая отображает:

- сколько пациентов записалось через их систему;
- сколько отзывов было оставлено на сайте;

- сколько практикующих врачей существует в их базе;
- сколько клиник подключено к сервису

Статистика представлена на рисунке 8.

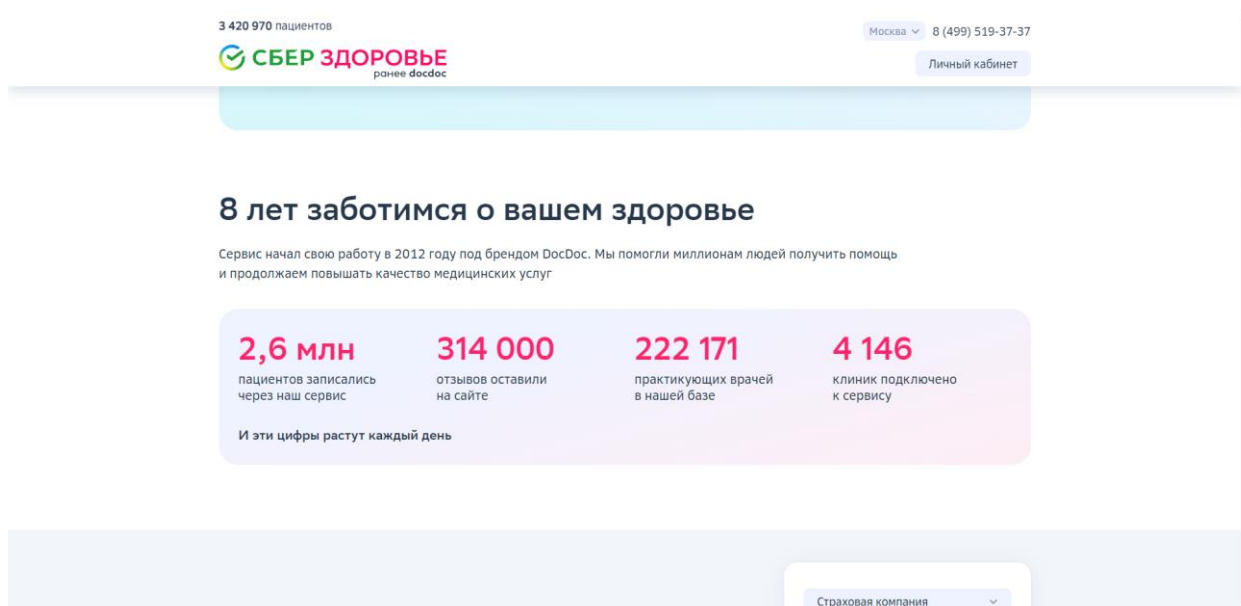


Рисунок 8 – Статистика системы «DocDoc.ru»

Также в рассматриваемой системе можно выбрать врачей, которые входят в программу страхования.

Системы были выбраны на основании схожести их функционала.

1.2.3 Результаты сравнительного анализа систем-аналогов

Сравнительный анализ систем-аналогов производился по функционалу, который необходим человеку для получения медицинской услуги или полноценной информации о медицинских учреждениях города, а также их контактные данные.

Существуют функциональные возможности, которые присущи всем системам, такие как:

- запись к врачу;
- расписание врачей;
- наличие сайта;
- просмотр всех организаций города.

Далее можно найти следующие различия, например, стоимость медицинских услуг не представлена в системе «Кубань–online». Данная

функция должна быть в каждой системе, чтобы пользователь мог узнать цену услуги.

Отображение медицинских организаций отсутствует в системах «Электронная регистратура», «MEDODS» и «DocDoc.ru». Для того, чтобы пользователь не тратил свое время на поиски в интернете местоположения больниц, было бы лучше, если все было в одном месте и в пределах одной системы.

Также, оставление отзыва о полученной услуге является не менее важной возможностью, так как организации, как и их представители и сотрудники должны знать о своих, как преимуществах, так и недостатках, которые следует доработать, чтобы исключить негативные отзывы об услугах. К сожалению, эта возможность отсутствует в системах «К–Врачу», «Кубань–online» и «MEDODS».

Помимо вышеизложенного существуют онлайн-консультации, которые позволяют пользователям получить ответы на вопросы, не выходя из дома. Онлайн-консультации есть в таких системах, как «Электронная регистратура», «Кубань–online» и «DocDoc.ru».

Сравнение систем представлено в таблице 4. В данной таблице присутствуют следующие обозначения:

- 1) «++» – присутствует в системе и удобно в использовании;
- 2) «+-» – присутствует в системе и неудобно в использовании;
- 3) «--» – отсутствует в системе.

Они необходимы для того, чтобы наглядно увидеть, насколько система является удобной в использовании.

Таблица 4 – Сравнительный анализ

Наименование	К–Врачу	Электронная регистратура	Кубань– online	MEDODS	DocDoc.ru
Запись к врачу	++	+-	+-	+-	++

Продолжение таблицы 4

Наименование	К–Врачу	Электронная регистратура	Кубань– online	MEDODS	DocDoc.ru
Запись на медицинскую услугу	++	++	++	+-	++
Расписание врачей	+-	++	++	++	+-
Просмотр всех организаций города	++	+-	++	++	+-
Стоимость медицинских услуг	+-	++	--	++	++
Подбор ближайшего медицинского учреждения	--	--	--	--	--
Отображение медицинских организаций на карте	++	--	++	--	--
Подбор специалиста по симптомам	--	--	--	--	--
Возможность оставить отзыв о медицинской услуге	--	+-	--	--	++
Онлайн консультация	--	++	+-	--	++
Возможность записей в регионах	++	+-	+-	+-	++

Окончание таблицы 4

Наименование	К–Врачу	Электронная регистратура	Кубань– online	MEDODS	DocDoc.ru
Сайт	+-	+-	++	+-	+-
Приложение	+-	--	--	+-	++

Исходя из просмотренных систем и их функционалов, необходимо реализовать такие возможности, как подбор ближайшего медицинского учреждения и поиск специалиста по симптомам. Пользователь сможет ввести свой адрес и увидеть какие организации находятся к нему ближе всего и выбрать наиболее подходящую из представленных системой. Если же пациент не знает к какому врачу идти на прием, то в этом ему сможет помочь поиск специалиста по симптомам. Для этого будет необходимо лишь ввести то, что тревожит (например, насморк, кашель, боль в суставах и т.д.), а система в свою очередь по введенным данным подберет наиболее подходящего специалиста. Таблица функционала, который предположительно будет реализован в данном дипломном проекте представлен в таблице 5. В ней под базовым функционалом подразумевается тот, который есть в большинстве систем, а новый – предположительно будет реализован в web-портале медицинских услуг.

Таблица 5 – Функционал web-портала медицинских услуг

№ п/п	Наименование функции	Портал медицинских услуг	Базовый функционал/ новый функционал
1	Запись к врачу	+	Базовый
2	Запись на медицинскую услугу	+	Базовый
3	Расписание врачей	+	Базовый
4	Просмотр всех организаций города	+	Базовый

Продолжение таблицы 5

№ п/п	Наименование функции	Портал медицинских услуг	Базовый функционал/ новый функционал
5	Стоимость медицинских услуг	+	Базовый
6	Подбор ближайшего медицинского учреждения	+	Новый
7	Отображение медицинских организаций на карте	+	Базовый
8	Подбор специалиста по симптомам	+	Новый
9	Возможность оставить отзыв о медицинской услуге	+	Базовый
10	Онлайн консультация	+	Базовый
11	Возможность записей в регионах	+	Базовый
12	Сайт	+	Базовый

Данный выше функционал может быть расширен при желании разработчика.

1.3 Постановка задачи на разработку

1.3.1 Цели разработки и назначение

Целью разработки является создание web-портала для поиска и заказа медицинских услуг, который должен обладать не только базовым функционалом, который встречается во многих системах-аналогах, но и иметь что-то новое для большей заинтересованности в ней пользователей.

1.3.2 Классы пользователей

Классы пользователей системы можно подразделить на внутренние и внешние. К внешним будут относиться пользователи и медицинские

работники, а к внутренним системный администратор и представитель медицинской организации.

Под пользователем понимается любой зарегистрированный в системе человек, который хочет получить медицинскую услугу. Все, что необходимо для регистрации это доступ к интернету и личные данные человека.

Для медицинского работника доступен просмотр своего расписание и записи на его услуги. Помимо этого, он может проводить онлайн-консультации пациентам.

Системный администратор может предоставлять доступ представителю медицинской организации для того, чтобы он мог вводить все необходимые данные о медицинском учреждении и поддерживать их актуальность. Также системный администратор может

1.3.3 Перечень автоматизируемых функций

Web-портал для поиска и заказа медицинских услуг автоматизирует следующие функции:

- запись на медицинские услуги;
- регистрация медицинского учреждения;
- внесение данных о медицинском учреждении;
- изменение данных о медицинском учреждении;
- поддержание актуальности данных о медицинском учреждении;
- онлайн-консультации врачей;
- формирование документов и ведение картотеки;
- анализ данных;
- составление отчетов по запросам.

Описанные выше функции помогут пользователям работать с данными системы намного быстрее. Не будет необходимости в самостоятельном сборе информации.

1.4 Планирование и функционально-стоимостный анализ процесса разработки

1.4.1 Основы функционально-стоимостного анализа

Под функционально-стоимостным анализом понимается метод системного исследования функции отдельного изделия или определенного производственно-хозяйственного процесса, или же управленческой структуры, направленный на минимизацию затрат в сферах проектирования, освоения производства, сбыта, промышленного и бытового потребления при высоком качестве, предельной полезности и долговечности [5].

Основные задачи функционально-стоимостного анализа:

- достижение оптимального соотношения между полезным эффектом объекта и совокупными затратами за его жизненный цикл;
- нахождение совершенно новых технических решений за счет применения функционального подхода;
- снижение расхода различных видов ресурсов по стадиям жизненного цикла объекта за счет ликвидации или сокращения вспомогательных и вредных (ненужных) функций объекта.

Основные особенности проведения функционально-стоимостного анализа:

- объектом анализа может быть любая система, ее подсистемы или элементы, по которым можно количественно выразить полезный эффект их функционирования по назначению;
- глобальным критерием функционально-стоимостного анализа является максимум полезного эффекта объекта на единицу совокупных затрат ресурсов за его жизненный цикл;
- анализируется оптимальность элементов полезного эффекта и совокупных затрат по объекту;
- при проведении функционально-стоимостного анализа прежде всего устанавливается целесообразность функций, которые должен выполнять проектируемый объект в конкретных условиях [6].

С помощью функционально-стоимостного анализа можно решить такие задачи как, определение действительной стоимости, идентификация работ, которые стоят больше всего (те, которые должны быть улучшены в первую очередь) и т.д.

1.4.2 Планирование процесса разработки

Для планирования процесса разработки были созданы диаграммы бизнес-процессов в нотации IDEF0 в программной среде BPwin.

Контекстная диаграмма представлена на рисунке 9.

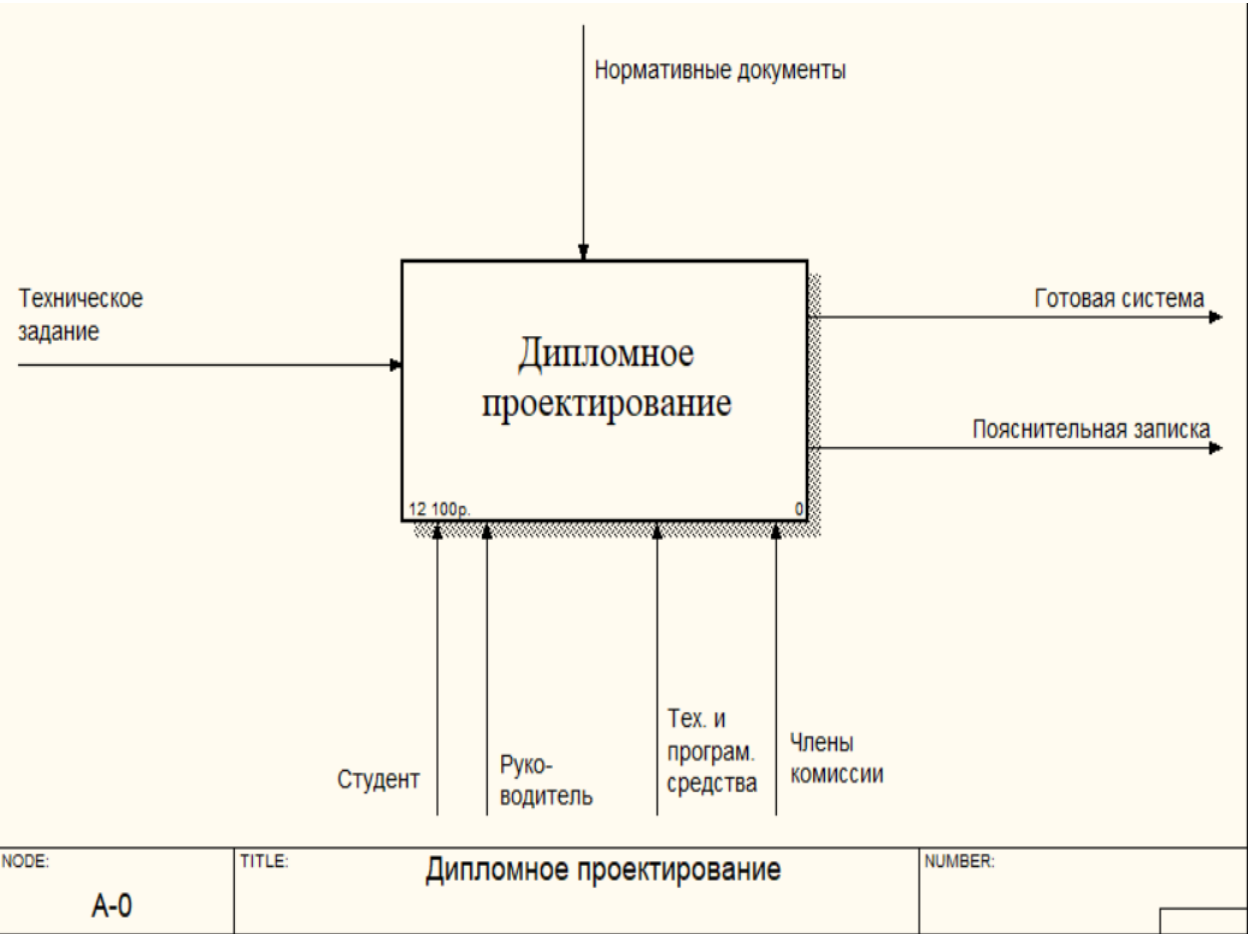


Рисунок 9 – Контекстная диаграмма

Диаграмма декомпозиции представлена на рисунке 10.

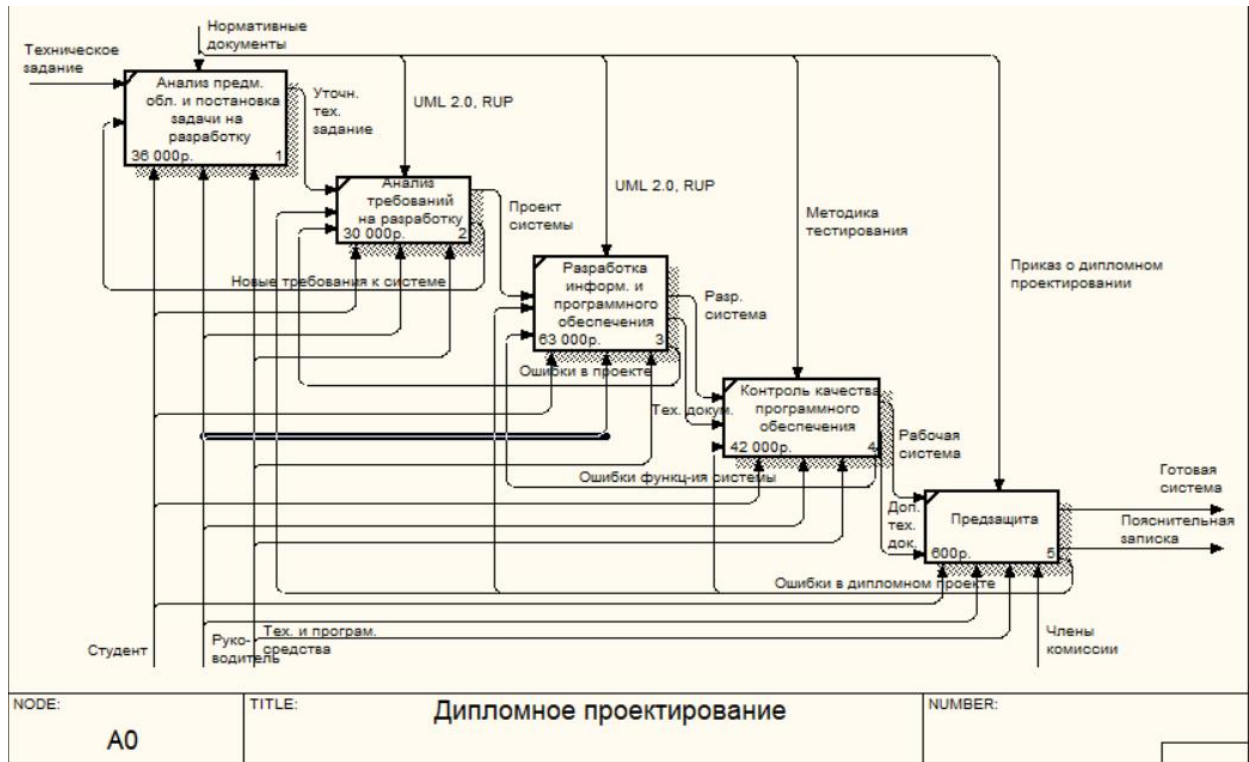


Рисунок 10 – Диаграмма декомпозиции

Методология IDEF0 предписывает построение иерархической системы диаграмм, т.е. единичных описаний фрагментов системы. Сначала проводится описание системы в целом и ее взаимодействия с окружающим миром (контекстная диаграмма), после чего проводится функциональная декомпозиция – система разбивается на подсистемы и каждая подсистема описывается отдельно (диаграммы декомпозиции). Затем каждая подсистема разбивается на более мелкие и так далее до достижения нужной степени подробности.

1.4.3 Анализ временных и финансовых затрат

Анализ временных и финансовых затрат необходим для реализации каждого проекта и является неотъемлемой его частью. Его осуществляет финансово-экономическая служба и используется он для того, чтобы проанализировать и выявить как денежные, так и временные отклонения.

Был проведен анализ временных затрат на проект, который приведен в таблице 6.

Таблица 6 – Анализ временных затрат на проект

№ п/п	Наименование этапа работы	Временной период
1	Анализ предметной области и постановка задачи на разработку	01.04.2021-18.04.2021
2	Анализ требований на разработку	19.04.2021-30.04.2021
3	Разработка информационного и программного обеспечения	01.05.2021-31.05.2021
4	Контроль качества программного обеспечения	01.06.2021-18.06.2021

Был проведен анализ финансовых затрат на проект, который приведен в таблице 7.

Таблица 7 – Анализ финансовых затрат на проект

№ п/п	Наименование этапа работы	Время	Тариф	Потрачено финансов
1	Анализ предметной области и постановка задачи на разработку	120 ч	300 руб/час	36000 руб
2	Анализ требований на разработку	100 ч	300 руб/час	30000 руб
3	Разработка информационного и программного обеспечения	210 ч	300 руб/час	63000 руб
4	Контроль качества программного обеспечения	140 ч	300 руб/час	42000 руб
5	Предзащита	2 ч	300 руб/час	600 руб

В итоге на разработку web-портала для поиска и заказа медицинских услуг было затрачено 171600 руб.

2 Анализ требований на разработку

2.1 Функциональные требования

Web-портал для поиска и заказа медицинских услуг должен быть удобным в использовании, так как он ориентирован на различные классы пользователей.

Для того, чтобы функционал web-портала был лучше воспринят, были созданы различные диаграммы с помощью унифицированного языка моделирования UML (Unified Modeling Language).

Функциональные возможности web-портала представлены на диаграмме вариантов использования, которая приведена на рисунке 11.

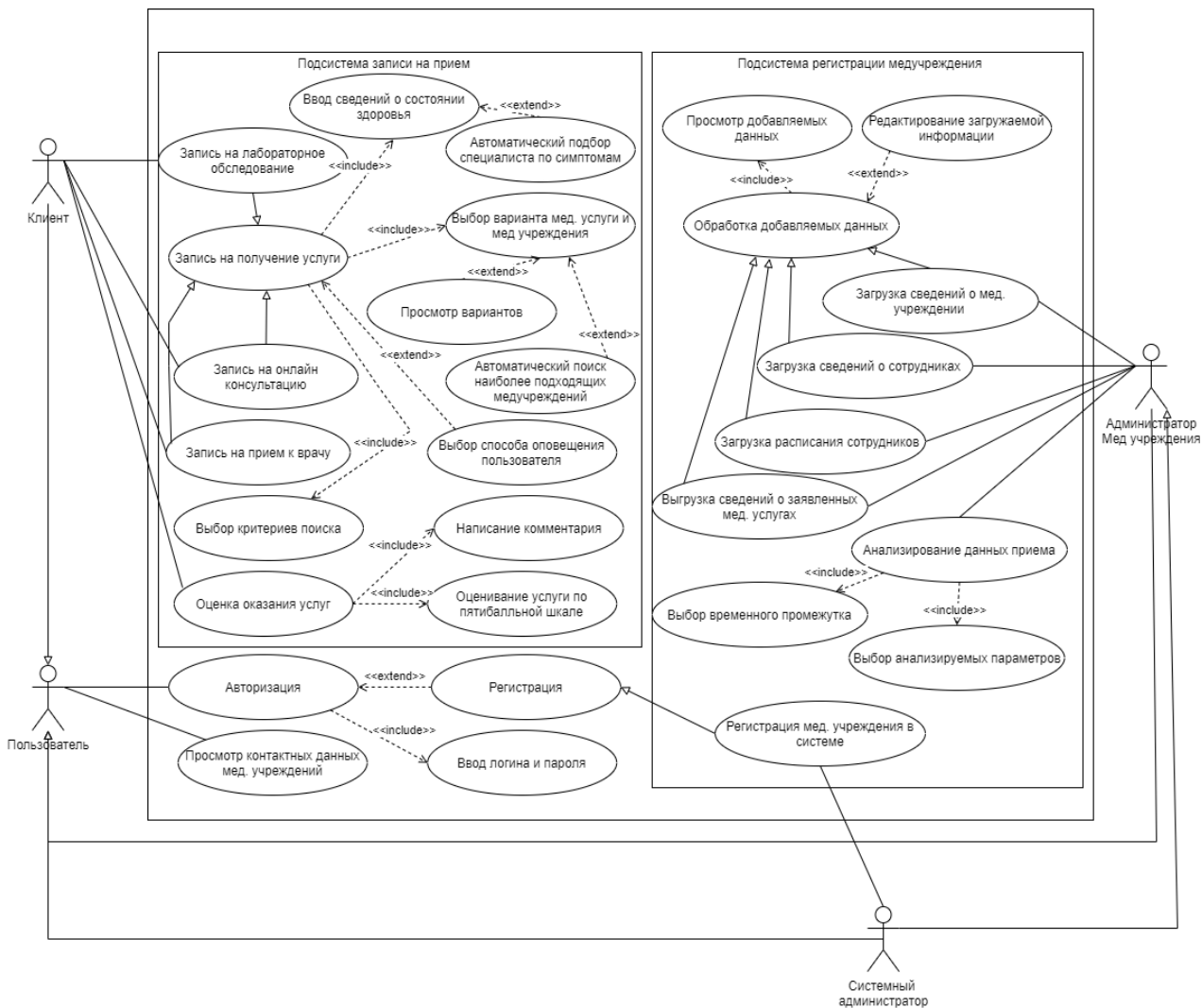


Рисунок 11 – Диаграмма вариантов использования

Краткое описание каждого варианта использования представлены в таблице 8.

Таблица 8 – Краткое описание вариантов использования

Вариант использования	Краткое описание
Запись на получение услуги	Пользователь может записаться на получение любой медицинской услуги
Запись на лабораторное обследование	Пользователь может записаться на любое лабораторное обследование
Запись на прием к врачу	Пользователь может записаться на прием к врачу
Запись на онлайн-консультацию	Пользователь может записаться на онлайн-консультацию необходимого ему врача
Ввод сведений о состоянии здоровья	Пользователь вводит сведения о своем состоянии здоровья в систему
Автоматический подбор специалиста по симптомам	Пользователь вводит имеющиеся у него симптомы, система, в свою очередь, должна подобрать специалиста, который сможет помочь
Выбор варианта медицинской услуги и медицинского учреждения	Пользователь может выбрать любую медицинскую услугу и медицинское учреждение
Просмотр вариантов	Пользователь может просмотреть весь список медицинских учреждений и медицинских услуг
Автоматический поиск наиболее подходящих медучреждений	По желанию пользователя система может автоматически подобрать наиболее подходящее медицинское учреждение, которое находится неподалеку от места проживания пользователя
Выбор способа оповещения пользователя	Пользователь может выбрать наиболее удобный для себя способ оповещения
Выбор критериев поиска	Пользователь может выбрать необходимые для него критерии поиска

Продолжение таблицы 8

Вариант использования	Краткое описание
Оценка оказания услуг	Пользователь может оценить полученные медицинские услуги
Написание комментария	Пользователь может оставить комментарий о полученной услуге
Оценивание услуги по пятибалльной шкале	Пользователь может оценить полученную услугу по пятибалльной шкале, от 1 до 5
Авторизация	Авторизация пользователей в системе
Регистрация	Регистрация пользователей в системе
Ввод логина и пароля	Для входа в систему зарегистрированному пользователю необходимо ввести логин и пароль
Просмотр контактных данных медицинских учреждений	Пользователь может просмотреть все контактные данные каждого медицинского учреждения
Обработка добавляемых данных	Администратор может обрабатывать данные, которые добавляет в систему
Просмотр добавляемых данных	Администратор может просмотреть данные, которые он добавил
Редактирование загружаемой информации	Администратор может отредактировать данные, которые были им добавлены
Загрузка сведений о медицинском учреждении	Администратор загружает сведения о своем медицинском учреждении
Загрузка сведений о сотрудниках	Администратор может загружать сведения о сотрудниках медицинского учреждения
Загрузка расписания сотрудников	Администратор может загрузить расписание сотрудников медицинского учреждения

Окончание таблицы 8

Вариант использования	Краткое описание
Выгрузка сведений о заявленных медицинских услугах	Администратор может выгрузить данные о записях пользователей на медицинские услуги их медицинского учреждения
Анализирование данных приема	Администратор может проанализировать данные с помощью системы
Выбор временного промежутка	Администратор может выбрать промежуток времени, за который необходимо проанализировать данные
Выбор анализируемых параметров	Администратор может выбрать параметры, по которым необходимо проанализировать данные
Регистрация медицинского учреждения в системе	Системный администратор может зарегистрировать в системе медицинское учреждение для дальнейшей работы с ним администратора

Диаграмма деятельности представляет собой поведенческую диаграмму, которая является более детальной визуализацией конкретного варианта использования. Она иллюстрирует не что должна делать система, а как, показывая последовательность действий для достижения необходимой пользователю цели. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчиненных элементов — вложенных видов деятельности и отдельных действий, соединенных между собой потоками, которые идут от выходов одного узла ко входам другого.

Взаимодействие пользователя с системой в результате выполнения варианта использования «Запись на получение медицинской услуги» представлена на диаграмме деятельности, которая приведена на рисунке 12.

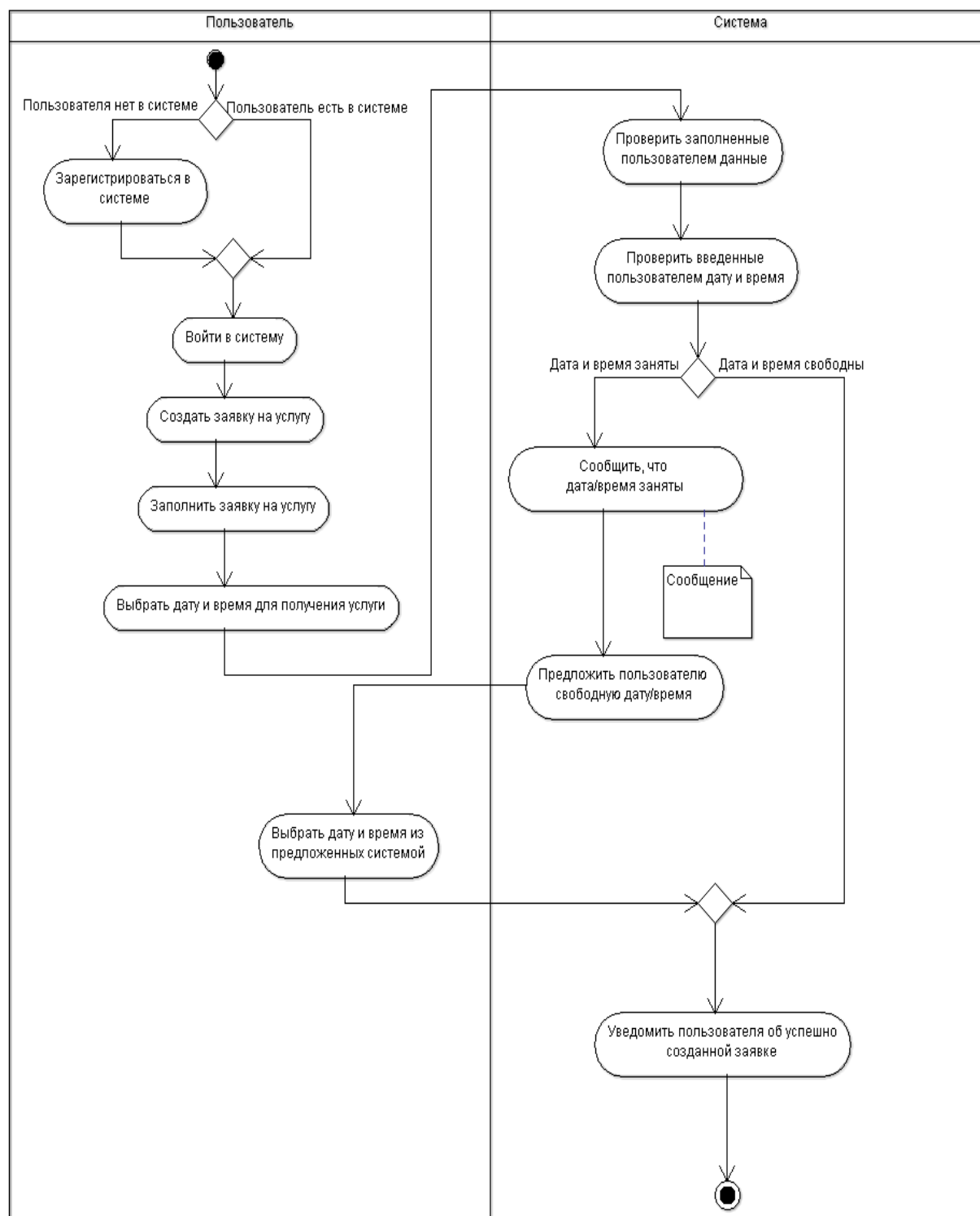


Рисунок 12 – Диаграмма деятельности для варианта использования «Запись на получение медицинской услуги»

Описание диаграммы деятельности для варианта использования «Запись на получение медицинской услуги» представлено в приложении А, таблицы А.1 – А.11.

Взаимодействие пользователя с системой в результате выполнения варианта использования «Автоматический подбор специалиста по симптомам» представлена на диаграмме деятельности, которая приведена на рисунке 13.

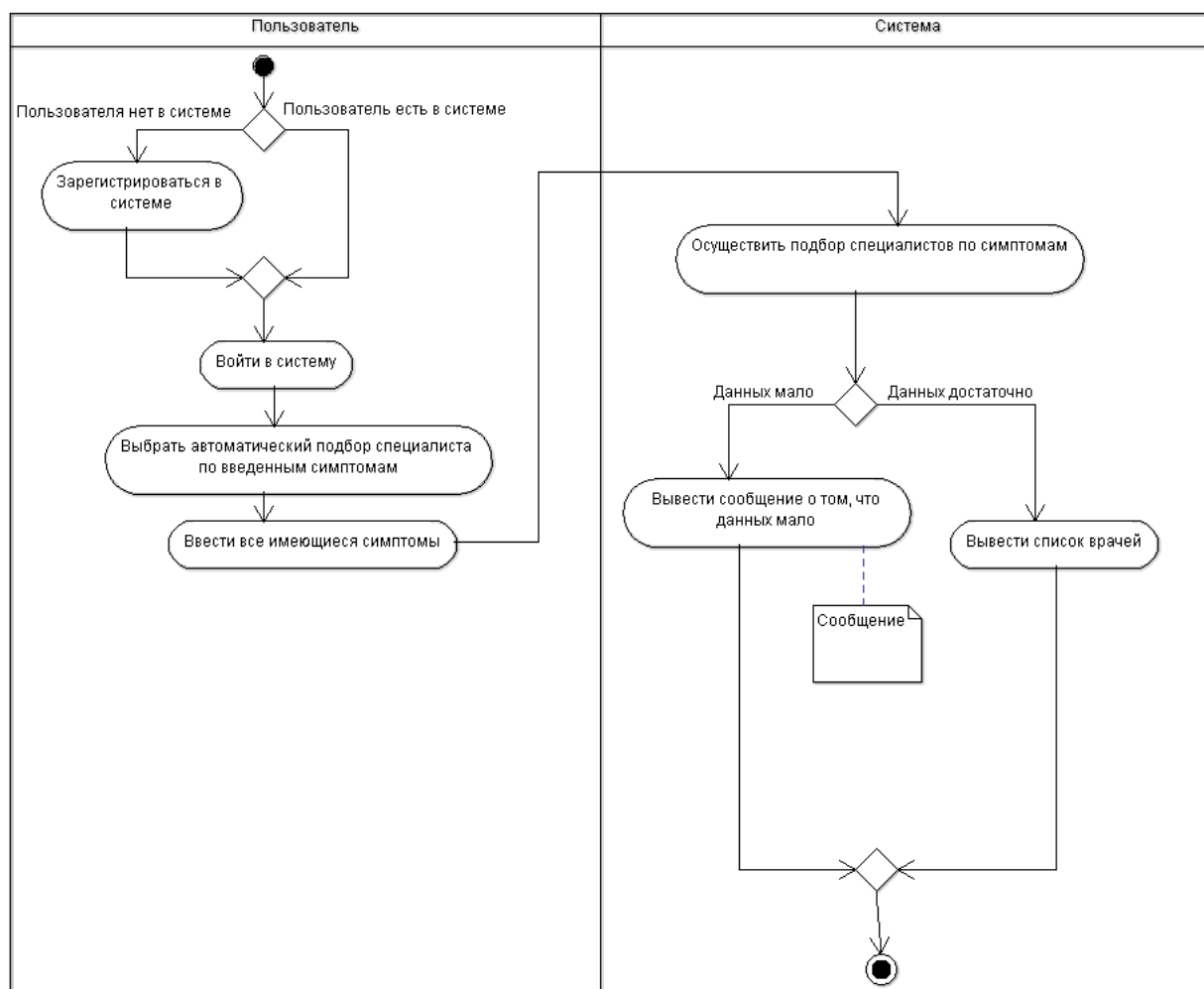


Рисунок 13 – Диаграмма деятельности для варианта использования «Автоматический подбор специалиста по симптомам»

Описание диаграммы деятельности для варианта использования «Автоматический подбор специалиста по симптомам» представлено в приложении А в таблицах А.12 – А.18.

Взаимодействие пользователя с системой в результате выполнения варианта использования «Автоматический поиск наиболее подходящих медучреждений» представлена на диаграмме деятельности, которая приведена на рисунке 14.

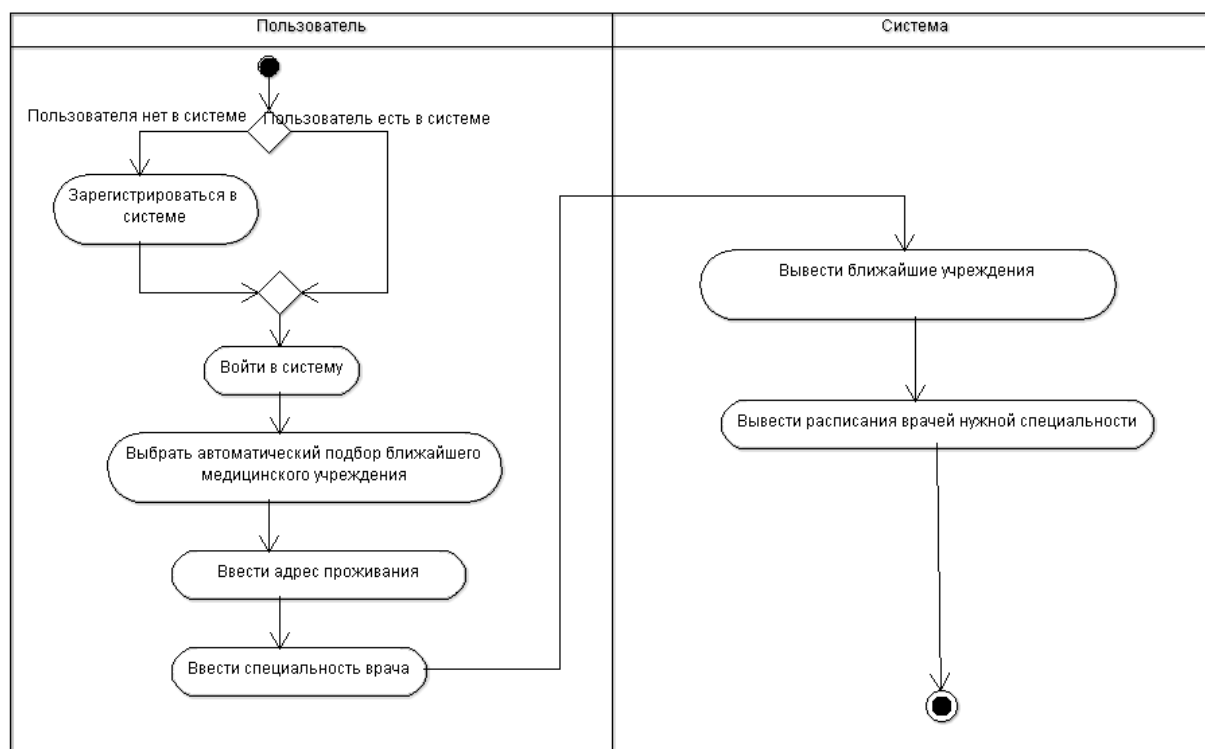


Рисунок 14 - Диаграмма деятельности для варианта использования «Автоматический поиск наиболее подходящих медучреждений»

Описание диаграммы деятельности для варианта использования «Автоматический поиск наиболее подходящих медучреждений» представлено в приложении А в таблицах А.19 – А.25.

Остальные взаимодействия пользователя с системой в результате выполнения вариантов использования не требует подробного описания.

2.2 Построение концептуальной модели

Концептуальная модель служит для описания обобщенного объекта, для которой необходимо построить модель и где не должно присутствовать описание отдельных деталей.

В виде концептуальной модели будет представлена диаграмма классов, которая приведена на рисунке 15.

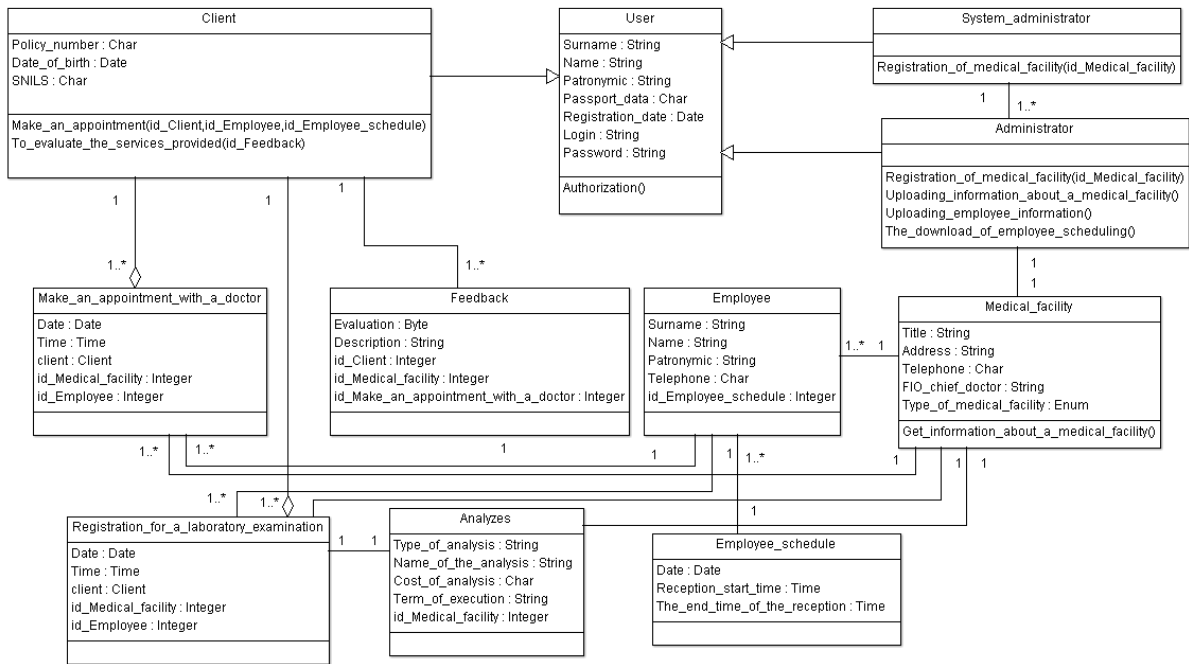


Рисунок 15 - Диаграмма классов концептуальной модели

Подробное описание диаграммы классов представлено в приложении Б в таблицах Б.1 – Б.11.

Класс представляет собой описание набора объектов с одинаковыми атрибутами, связями, операциями и семантикой. Графически он изображается в виде прямоугольника, который разделен на три части с помощью горизонтальных линий (имя класса, атрибуты класса, операции класса).

2.3 Требования к обобщенной архитектуре приложения

Обобщенная архитектура необходима для того, чтобы по мере усложнения приложения, оно не теряло свою гибкость, управляемость и не становилась дороже с каждым незначительным изменением. Существует четыре самых известных архитектуры, такие как: многоуровневая, многослойная, сервис-ориентированная и микросервисная архитектура [8].

Архитектура приложения представлена в виде диаграммы развертывания на рисунке 16.

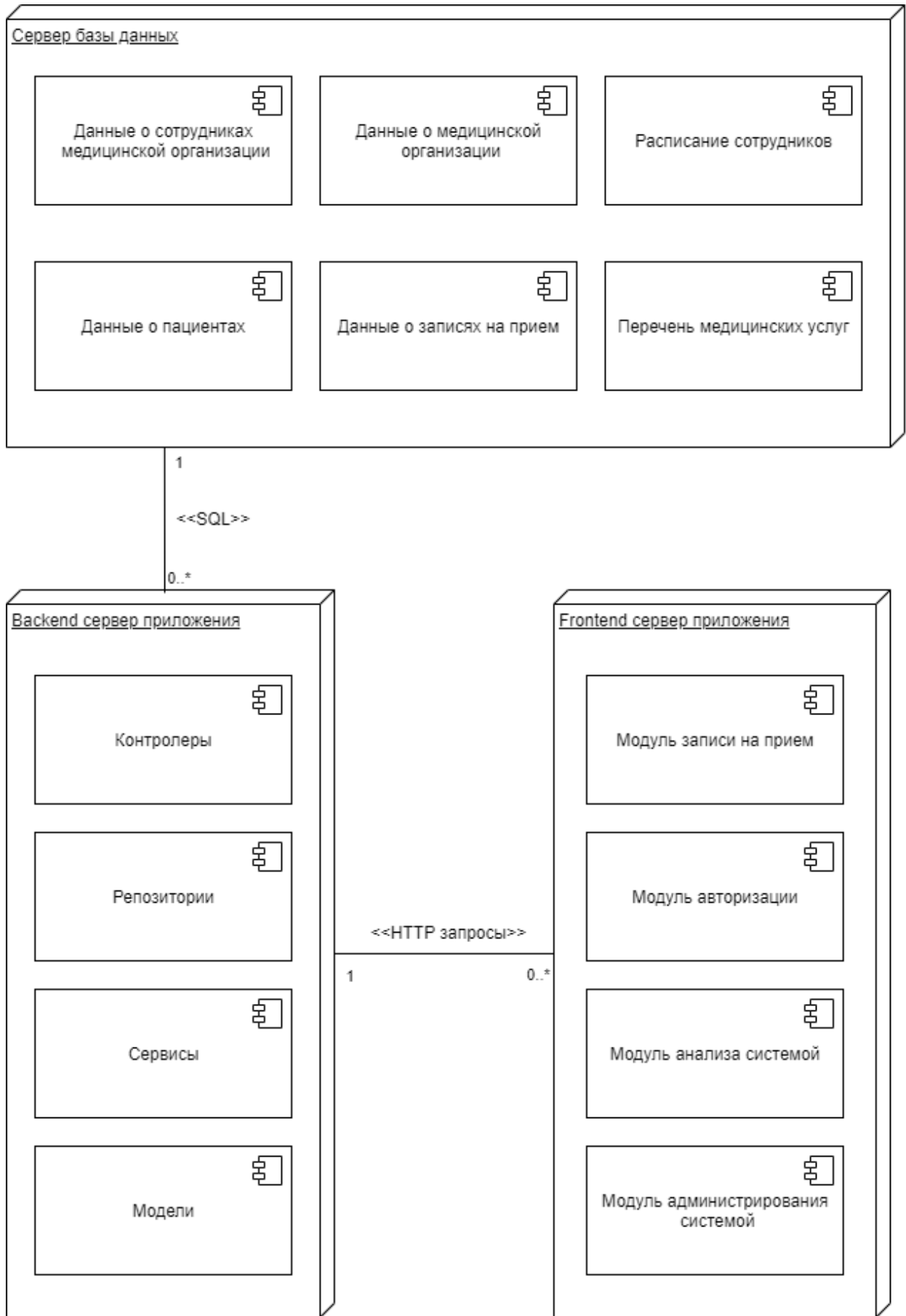


Рисунок 16 – Диаграмма развертывания

На выше представленной диаграмме изображены основные компоненты разрабатываемой системы:

- сервер базы данных;
- backend сервер приложения;
- frontend сервер приложения.

Сервер базы данных необходим для обработки SQL запросов, здесь хранятся данные, производимые в процессе работы системы. Backend сервер приложения обрабатывает запросы клиента в http формате, здесь находится основная логика работы приложения. Frontend сервер приложения обеспечивает отображение и работу с данными в удобном для пользователя интерфейсом.

У каждой архитектуры существуют как плюсы, так и минусы. Далее они будут рассмотрены подробнее.

Многослойная архитектура представляет собой принцип разделения ответственностей, то есть, исходя из самого названия, архитектура состоит из следующих слоев, из которых каждый выполняет свои определенные обязанности: слой представления, бизнес-логики и передачи данных.

К достоинствам относится: простота реализации, защита одного слоя от изменений в другом слое, повышает управляемость программы, ответственность между слоями разделенная.

К недостаткам относится: отсутствие большой масштабируемости, наличие монолитной системы, которая усложняет внесение различных изменений (модификаций), данные должны проходить по каждому слою.

Многоуровневая архитектура использует принцип клиент-сервер. Она может иметь один, два и более уровней. Уровни служат для разделения ответственности между поставщиком данных и их потребителем.

К достоинствам относится: независимость уровней друг от друга (модификации на одном уровне никак не отразятся на другом уровне), существуют специальные генераторы шаблонов.

К недостаткам относится: низкая скорость работы.

Сервис-ориентированная архитектура представляет собой модель, состоящую из компонентов и приложений, связанных сервисами.

К достоинствам относится: легкая масштабируемость, высокая производительность приложения.

К недостаткам относится: усложненная отладка, сложность в нахождении причины сбоев (ошибок).

Микросервисная архитектура позволяет разрабатывать приложение как набор сервисов, которые работают в собственных процессах и взаимосвязаны с помощью легковесных механизмов.

К достоинствам относится: модульность, независимость сервиса и системы в случае сбоя, простота модификации, высокая модульность и масштабируемость, решение проблем с потоками данных.

К недостаткам относится: повышенный риск сбоя при обмене данных между серверами, трудное управление большим количеством сервисов, возможность задержки сети, необходимость в комплексном тестировании, большие затраты времени [9].

2.4 Обоснование выбора технологий и средств разработки

2.4.1 Обоснование выбора системы управления базами данных

Выбор системы управления базами данных (СУБД) представляет собой сложную задачу и является одним из важных этапов при разработке приложений баз данных [11]. В настоящее время практически ни одно учреждение или компания не обходится без систем управления базами данных. Они необходимы для составления различных видов отчетов, получения необходимой информации, анализа данных и их агрегацию [12].

По способу доступа к базе данных СУБД подразделяют на:

- 1) СУБД с централизованной архитектурой;
- 2) СУБД с архитектурой файл-сервер;
- 3) СУБД с архитектурой клиент-сервер;
- 4) СУБД с трехуровневой архитектурой: «тонкий клиент» – сервер приложений – сервер базы данных [13].

Для реализации web-портала для поиска и заказа медицинских услуг будет использована СУБД с архитектурой клиент-сервер. Существует

множество различных приложений для баз данных. Например: Microsoft SQL Server, MySQL, Microsoft Access, Oracle и др.

Для сравнительной характеристики были выбраны такие СУБД, как Microsoft SQL Server, MySQL и Microsoft Access. Сравнительная характеристика СУБД представлена в таблице 9.

Таблица 9 – Сравнительная характеристика СУБД

Характеристика	Microsoft SQL Server	Microsoft Access	MySQL
Фирма, производитель	Microsoft	Microsoft	Oracle
Год выпуска	2008	2007	2013
Поддерживаемые операционные системы	Windows/XP/7 и выше	Windows/XP/Vista/7 и выше	Кроссплатформенное ПО
Максимальное количество символов в имени поля	128	64	256
Минимальные системные требования	166 МГц, 64 Мб ОЗУ, 250 Мб	500 МГц, 256 Мб ОЗУ, 1,5 Гб	ОЗУ 256 Мб, HDD 2 Гбайт, Pentium II 233 МГц
Максимальный размер базы	16 ТБ	2 Гб за вычетом места, необходимого для системных объектов.	Неограниченный

Для разработки web-портала для поиска и заказа медицинских услуг была выбрана СУБД MySQL.

2.4.2 Обоснование выбора языковых и инструментальных средств

Язык программирования – формальная знаковая система, предназначенная для записи компьютерных программ. Язык

программирования определяет набор лексических, синтаксических и семантических правил, задающих внешний вид программы и действия, которые выполнит исполнитель (компьютер) под ее управлением [14].

Самыми распространенными и популярными языками программирования на данный момент являются: C#, C++ и JavaScript. Сравнительная таблица языков программирования приведена в таблице 10.

Таблица 10 – Сравнение языков программирования

Характеристика	C++	C#	JavaScript
Фирма	Borland	Microsoft	Symantec, Microsoft
Год создания	1984 г.	1998 г.	1995 г.
Поддержка библиотек	Средний уровень	Средний уровень	На высоком уровне
Интерфейс пользователя	Низкий уровень	Средний уровень	На высоком уровне
Кросс-платформенность	На высоком уровне	Низкий уровень	На высоком уровне
Скорость обработки	На высоком уровне	На высоком уровне	Средний уровень
Простота	Низкий уровень	Средний уровень	Средний уровень

В ходе сравнительной таблицы за основу разработки web-приложения был выбран язык JavaScript.

2.4.3 Обоснование выбора фреймворка для разработки приложения

React (иногда React.js или ReactJS) – JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов. React может использоваться для разработки одностраничных и мобильных приложений. Его цель – предоставить высокую скорость, простоту и масштабируемость. В качестве библиотеки для разработки пользовательских интерфейсов React часто используется с другими библиотеками, такими как MobX, Redux и

GraphQL. React был создан Джорданом Валке, разработчиком программного обеспечения из Facebook. На него оказал влияние ХНР – компонентный HTML-фреймворк для PHP. Впервые React использовался в новостной ленте Facebook в 2011 году и позже в ленте Instagram в 2012 году. Исходный код React открыт в мае 2013 года на конференции «JSConf US» [15].

React Native анонсирован на конференции Facebook «React.js Conf» в феврале 2015 года, а исходный код открыт в марте 2015 года. Он позволяет разрабатывать нативные Android, iOS и UWP-приложения с использованием React.

18 апреля 2017 года Facebook анонсировал React Fiber, переписанную и оптимизированную версию React. React Fiber станет основой разработки всех будущих функций и улучшений.

Одна из ключевых особенностей React – универсальность. Эту библиотеку можно использовать на сервере и на мобильных платформах с помощью React Native. Это принцип Learn Once, Write Anywhere или «Научитесь один раз, пишите, где угодно».

Еще одна важная особенность библиотеки – декларативность. С помощью React разработчик описывает, как компоненты интерфейса выглядят в разных состояниях. Декларативный подход сокращает код и делает его понятным.

React основан на компонентах, это еще одна ключевая особенность библиотеки. Каждый компонент возвращает часть пользовательского интерфейса со своим состоянием. Объединяя компоненты, программист создает завершенный интерфейс web-приложения.

Важная особенность React – использование JSX. Это расширение синтаксиса JavaScript, которое удобно использовать для описания интерфейса. JSX похож на HTML, тем не менее это всё-таки JavaScript.

JSX позволяет писать JavaScript-код с помощью готовых компонентов, которые практически полностью повторяют HTML. Это упрощает разработку.

К важным особенностям React относится использование виртуального DOM (Virtual DOM). Виртуальный DOM – объект, в котором хранится

информация о состоянии интерфейса. При изменении состояния, например, после отправки формы или нажатия кнопки, React рассчитывает разницу между старым и новым состоянием. После этого библиотека отрисовывает новое состояние. Использование виртуального DOM позволяет библиотеке эффективно обновлять реальный DOM [16].

2.5 Требования к аппаратно-программному обеспечению

2.5.1 Требования к аппаратно-техническому обеспечению

Минимальные аппаратно-технические требования представлены в таблице 11.

Таблица 11 – Минимальные аппаратно-технические требования

Компонент	Центральный процессор	Оперативная память	Пространство на жестком диске	Платформа
frontend сервер (Node js)	1 ГГц	512 Мб	20 Мб	Microsoft Windows, Linux x32/x64
backend server (JavaScript)	Pentium 2 266 МГц	128 Мб	124 МБ для JRE; 2 МБ для обновления Java	Microsoft Windows, Linux x32/x64
Сервер БД (MySQL)	-	178 Мб	-	Microsoft Windows, Linux x32/x64

Помимо вышеперечисленных требований, также необходим доступ к сети Internet.

2.5.2 Требования к системному программному обеспечению

К требованиям к системному программному обеспечению относятся:

- 1) надежность;

- 2) гарантия выполнения всех функциональных требований;
- 3) должен присутствовать состав технических средств;
- 4) должна быть выполнена проверка на совместимость программных средств;
- 5) универсальность;
- 6) легкость в понимании и использовании;
- 7) ОС, обеспечивающая работу с файлами;
- 8) ОС, обеспечивающая пользовательский интерфейс.

А для использования приложением необходимо наличие браузера.

3 Разработка информационного и программного обеспечения web-портала

3.1 Разработка архитектуры web-портала

Для разработки архитектуры приложения был выбран шаблон проектирования MVC (Model View Controller), который позволяет разделить приложение на 3 части:

- model (Модель);
- view (вид или представление);
- controller (контроллер).

Модель предоставляет данные и реагирует на команды контроллера, изменяя свое состояние. Так как модель является постоянным хранилищем данных, которое используется во всей структуре, то она должна обеспечивать полноценный доступ к данным для их просмотра, записи, отбора и т.д. В разработке web-портала для поиска и заказа медицинских услуг в качестве модели выступает база данных MySQL, в которой создаются необходимые для работы таблицы, которые заполняются данными, используемые для корректной работы приложения.

Представление отвечает за отображение данных модели пользователю, реагируя на изменения модели. В web-портале для поиска и заказа медицинских услуг представлением выступает компонент, в котором генерируется и отображается HTML (Hyper Text Markup Language) код и представлен он в виде Node.js React.

Контроллер интерпретирует действия пользователя, оповещая модель о необходимости изменений, а также является единственной частью схемы, где необходимо взаимодействие пользователя. Контроллер не содержит никакой логики, кроме как сбора данных, а также может быть подключен только к одному представлению и одной модели. В качестве контроллера в web-портале для поиска и заказа медицинских услуг выступает Java Script [17].

Шаблон проектирования MVC – это шаблон программной архитектуры, построенный на основе сохранения представления данных отдельно от методов, которые взаимодействуют с данными.

Основная цель применения этой концепции состоит в отделении бизнес-логики (модели) от ее визуализации (представления, вида). За счет такого разделения повышается возможность повторного использования кода. Таким образом клиентское приложение (визуализация) не привязано к серверу, что позволяет разрабатывать различные клиенты под разные платформы и задачи [18].

Схема архитектуры, используемой в приложении, изображена на рисунке 17.

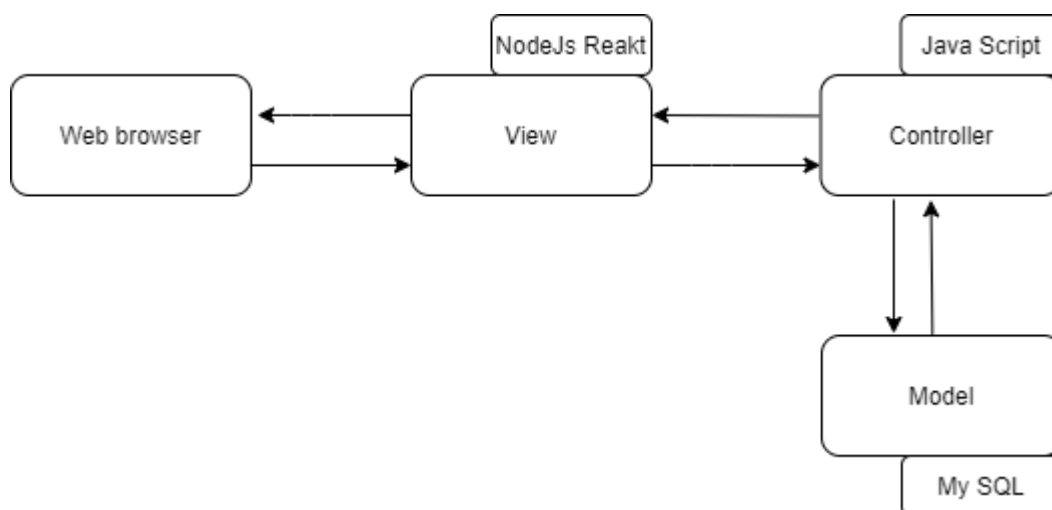


Рисунок 17 – Архитектура разрабатываемой системы

В схеме архитектуры разрабатываемой системы клиентом выступает Web-browser. Web-browser запрашивает у представления необходимую страницу. Представление формирует страницу, посылая при этом запросы на контроллер для получения актуальной информации из модели. Контроллер запрашивает данные у модели, формирует из них ответ и возвращает данный ответ представлению, которое в свою очередь предоставляет пользователю готовую страницу. Работая с представлением, пользователь также может отправлять запросы на изменение данных, которые будут обработаны контроллером и переданы в модель.

3.2 Разработка базы данных web-портала

База данных – совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных [12].

Разработанная база данных представлена на рисунке 18.

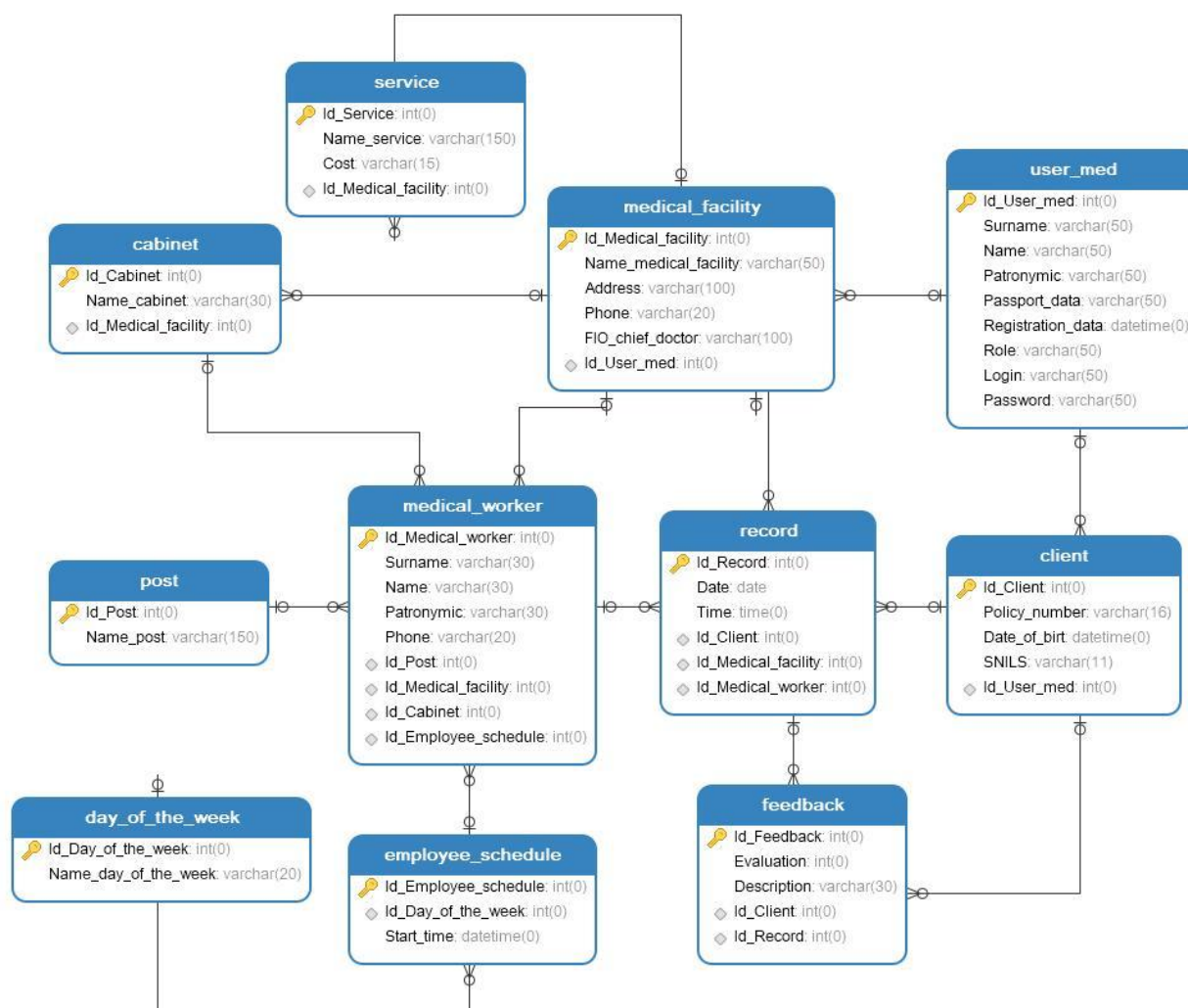


Рисунок 18 – Физическая схема базы данных

База данных состоит из 11 таблиц, каждая из которых описывается в виде перечисления полей. В каждом поле указывается: связь между таблицами (первичный и вторичный ключ), название поля, тип хранимых значений, в скобках указано максимальное количество возможных символов. Изображение ключа в таблице означает первичный ключ, а ромб используется в качестве обозначения вторичного ключа.

Первичный ключ – это поле, которое используется для обеспечения уникальности данных в таблице. Это означает, что значение (информация) в

поле первичного ключа в каждой строке (записи) таблицы может быть уникальным. Уникальность необходима для того, чтобы не было идентичных записей, и работа с базой данных для человека не составляла труда, так как будут отсутствовать различного рода неоднозначности, когда пользователь не знает к какой именно записи необходимо обратиться.

Вторичный ключ – это атрибут (набор атрибутов) одного отношения, который является ключом другого отношения. Он необходим для установления логических связей между отношениями (таблицами). Его присутствие в таблице не является обязательным, а его значение может быть не уникальным. Также вторичный ключ может принимать значение NULL.

Значение NULL означает отсутствие данных, констатирует, что значение неизвестно.

Подробное описание каждой сущности базы данных представлено в приложении В в таблицах В.1 – В.11.

База данных была создана вручную и привязана к WebStorm.

Код создания таблиц базы данных представлен в приложении Г.

3.3 Разработка программного обеспечения серверного приложения web-портала

Для разработки web-портала для поиска и заказа медицинских услуг в качестве прикладного программного интерфейса, который использует HTTP-запросы выступает Rest API (REpresentational State Transfer Application programming interface) [19].

Сервер создан с помощью фреймворка Express, написанный на JavaScript, также он использует модуль http.

Также к серверу был подключен MySQL, где находится база данных «medical_vkr». Подключение базы данных представлен в листинге 1.

Листинг 1 – Подключение базы данных

```
const mysql = require('mysql')

const sqlConfig = {
  host: "localhost",
  user: "root",
  password: 'root',
```

```

    database: 'medical_vkr'
  }
  const connection = mysql.createConnection(sqlConfig);
  connection.connect(err => {
    if (err) throw err;
    console.log("успешно соединено с базой данных");
  });

```

module.exports = connection;

В случае успешного подключения базы данных, будет выведено сообщение «успешно соединено с базой данных».

Также на сервере прописаны запросы к базе данных. Один из них необходим для получения всех данных о медицинских учреждениях, которые были прописаны вручную, как тестовые данные. Запрос к базе данных представлен в листинге 2.

Листинг 2 – Запрос к базе данных для медицинских учреждений

```

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

  const sql = "SELECT * FROM medical_facility";

  db.query(sql, function (err, data, fields) {
    if (err) {
      next(new Error('Ошибка сервера!'));
    }

    res.json({
      status: 200,
      data: data
    });

  });

}, function (err, req, res, next) {
  res.json({
    status: 501,
    message: err.message
  })
});

```

В случае ошибки будет выведено сообщение «ошибка сервера».

Запрос на получение данных из таблицы «post» представлен в листинге 3.

Листинг 3 – Запрос на получение данных из таблицы «post»

```

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

  const sql = "SELECT * FROM post";

  db.query(sql, function (err, data, fields) {
    if (err) {

```

```

        next(new Error('Ошибка сервера!'));
    }

    res.json({
        status: 200,
        data: data
    });

});

}, function (err, req, res, next) {
    res.json({
        status: 501,
        message: err.message
    })
});

```

На скриншотах, представленных выше «router.post» служит для создания запроса на ввод данных, которые передаются в указанное место и обрабатываются. В константу «sql» записывается запрос, где «SELECT» необходим для выбора из таблицы, «*» необходима для выбора всех данных, а не конкретных, «FROM» необходим для выбора таблицы, после него идет название таблицы, например, «post».

Остальные запросы, которые служат для получения данных из MySQL представлены таким же образом.

На листинге 4 представлена обработка логина пользователя.

Листинг 4 – Обработка логина пользователя

```

router.post('/login', function (req, res, next) {
    if (!req.body) {
        res.sendStatus(400);
    }

    let nickname = req.body.username;
    let password = req.body.password;
    const sql = "SELECT Id_User_med, Login, Password, Role FROM
user_med WHERE Login =?";
    let dataSql = [nickname];

    db.query(sql, dataSql, function (err, data, fields) {
        if (err) {
            next(new Error('Ошибка сервера!'));
        } else if (!data.length) {
            next(new Error('Введены некорректные данные!'));
        } else {
            bcrypt.compare(password,
data[0].password).then((result) => {
                if (result === true) {
                    const userData = {
                        id: data[0].id,

```

```

        nickname: data[0].nickname,
        id_role: data[0].id_role,
    };
    const token = jwt.sign(userData,
"23r34t33w4grthynjw5u6u45eye");
    userData.token = token;
    console.log(token);
    res.json({
        status: 200,
        message: 'access is allowed',
        user: userData
    })
    } else {
        next(new Error('Введены некорректные
данные!'));
    }
    })
    }
    });
}, function (err, req, res, next) {
    res.json({
        status: 501,
        message: err.message
    })
});

```

Сервер проверяет введенные пользователем данные. Если пользователем будут введены корректные данные, то он получит сообщение «access is allowed», что означает доступ разрешен и пользователь может продолжить работу с web-порталом. Иначе, если введены некорректные данные, то выведется соответствующее сообщение «введены некорректные данные» или же ошибка 501.

Код создания сервера представлен в приложении Д.

3.4 Разработка пользовательского интерфейса web-портала

Разработка пользовательского интерфейса является неотъемлемой частью web-портала, так как ориентировано оно именно на пользователя.

Пользовательский интерфейс – это совокупность средств и правил взаимодействия человека и компьютера.

Интерфейс главной страницы представлен на рисунках 19, 20.

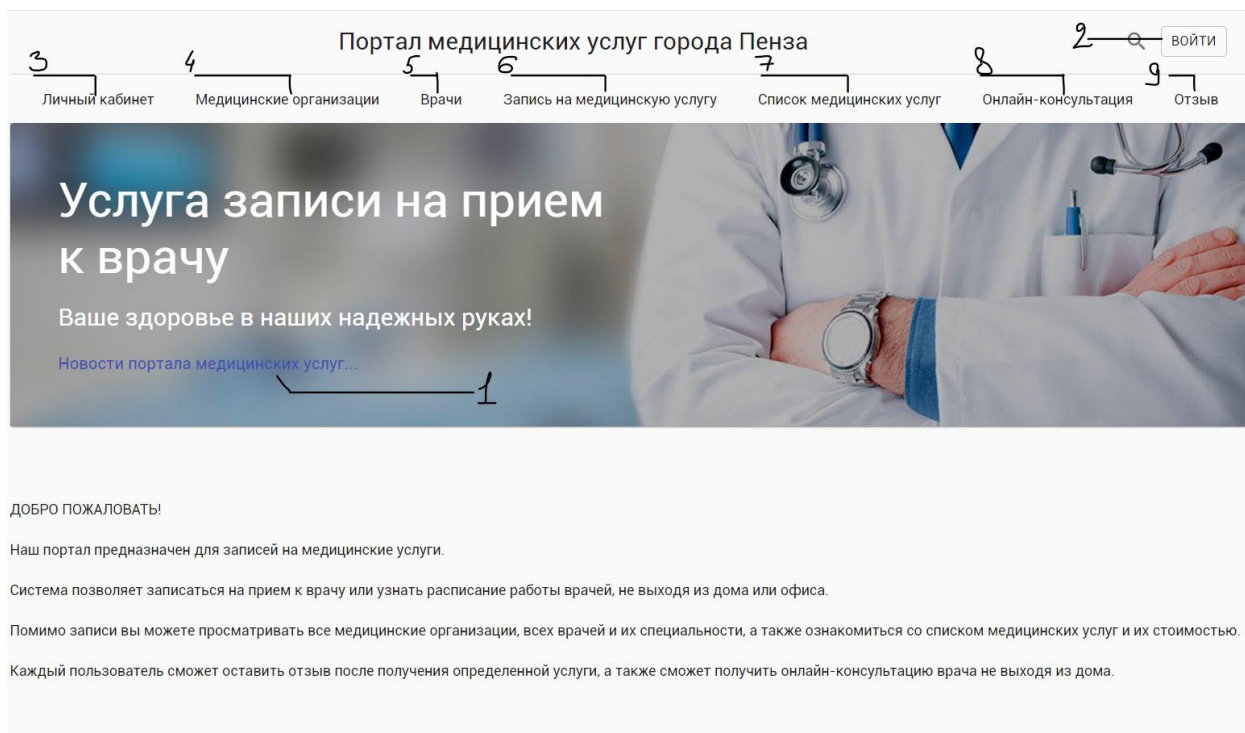


Рисунок 19 – Интерфейс главной страницы (часть 1)

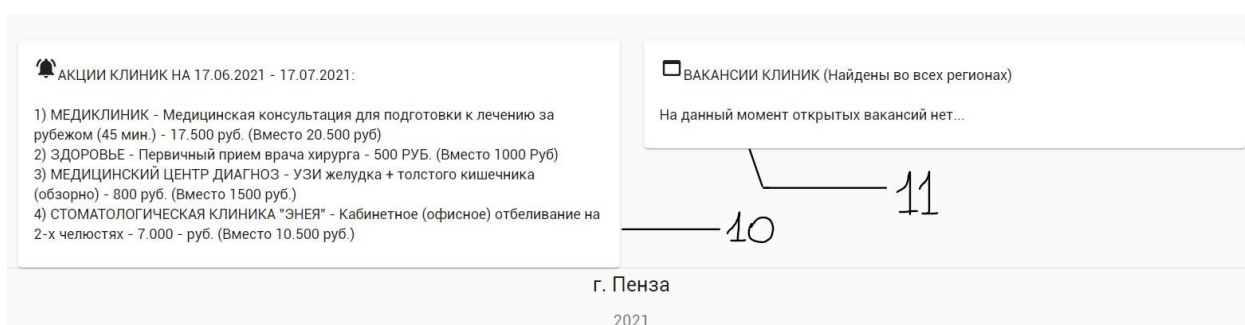


Рисунок 20 – Интерфейс главной страницы (часть 2)

Пункт 1 на рисунке 19 интерфейса главной страницы позволяет пользователю перейти на страницу с новостями web-портала медицинских услуг. Пункт 2 необходим для входа или регистрации пользователя, а также для создания нового медицинского учреждения.

Пункт 3 позволяет пользователю перейти в свой личный кабинет, где расположена его личная информация. При переходе в пункт 4, пользователь сможет увидеть все медицинские организации города, а также информацию о них.

Пункт 5 позволяет пользователю увидеть всех врачей и их специальности. В пункте 6 пользователь может записаться на необходимую ему медицинскую услугу.

В пункте 7 находится перечень всех медицинских услуг и их стоимость.

Пункт 8 позволяет пользователю получить онлайн-консультацию врача.

В пункте 9 пользователь может оставить отзыв о полученной медицинской услуге.

Пункт 10 отображает пользователю акции различных клиник, которые проходят в определенный период времени.

Пункт 11 предназначен для тех, кто ищет работу. Здесь размещаются все возможные свободные вакансии в медицинских учреждениях.

На рисунке 21 – 23 продемонстрирована запись пользователя к врачу на медицинскую услугу.

Рисунок 21 – Запись на медицинскую услугу (часть 1)

Рисунок 22 – Запись на медицинскую услугу (часть 2)

Рисунок 23 – Запись на медицинскую услугу (часть 3)

Если запись к врачу успешно подтверждена, то пользователь увидит сообщение на экране.

Код разработки клиентского интерфейса представлен в приложении Е.

4 Контроль качества программного обеспечения

4.1 Планирование процесса контроля качества программного обеспечения

Под качественным ПО (программное обеспечение) понимается ПО, которое не содержит различных ошибок, а также соответствует всем установленным требованиям. Основными требованиями являются:

- функциональность;
- надежность;
- мобильность;
- эффективность;
- удобство использования;
- сопровождение.

Под функциональностью понимается набор действий, которые решают и выполняют задачи пользователя. Они описаны в функциональных требованиях к ПО.

Под надежностью понимается такое требование, при котором в ходе работы ПО должно выполнять весь необходимый функционал за определенное время, при этом корректно завершать работу.

Под мобильностью понимается использование ПО на различных устройствах, не теряя при этом функциональность.

Под эффективностью понимается степень производительности программы с выделенными для нее ресурсами ОС (операционной системы).

Под удобством использования понимается легкость использования приложения пользователем.

Под сопровождением понимаются требования к модернизации приложения, исправление ошибок и обновление различных данных [20].

Все описанные выше показатели качества представлены на рисунке 24.



Рисунок 24 – Показатели качества

Планирование процесса контроля качества необходима для того, чтобы сотрудники и предприятие в целом были уверены, что все поставленные требования будут выполнены в срок, что поможет персоналу понять возможности улучшения.

Процесс контроля качества состоит из следующих этапов: определение требований, проектирование, кодирование, контроль исходного кода, анализ кода, конфигурационное управление, тестирование, управление релизами и интеграция продуктов.

4.1.1 Мероприятия по контролю качества программного обеспечения

Одной из самых важных техник контроля качества является тестирование. Тестирование включает в себя планирование работ, проектирование тестов, выполнение тестирования и анализ полученных результатов.

Существует 4 типа тестирования:

- функциональное;
- нефункциональное;
- структурное;
- тестирование изменений.

Функциональное тестирование, оно же тестирование методом «черного ящика», является одним из ключевых видов тестирования, задача которого – установить соответствие разработанного программного обеспечения исходным функциональным требованиям заказчика. То есть проведение функционального тестирования позволяет проверить способность информационной системы в определенных условиях решать задачи, нужные пользователям.

Нефункциональное тестирование проверяет насколько хорошо работает система. Он включает в себя такие виды, как: тестирование пользовательского интерфейса, тестирование удобства пользования, тестирование локализации, тестирование производительности, тестирование инсталляции и конфигурационное тестирование.

Структурное тестирование (или тестирование по принципу белого ящика) основывается на детальном изучении логики программы и подборе тестов, обеспечивающих максимально возможное число проверяемых операторов, логических ветвлений и условий. Данная методология тестирования предполагает выполнение каждого пути в программе.

Тестирование изменений необходимо для того, чтобы после различного рода обновлений, изменений, устранения различного рода ошибок, было проведено тестирование, которое поможет выявить, либо новые ошибки, либо, что не все ошибки были исправлены и в лучшем случае, что ошибок больше не осталось и программа функционирует стабильно и никаких нарушений не выявлено [21].

4.2 Тестирование программного обеспечения

Наиболее важными и необходимыми в разработке ПО видами тестирования на первоначальном этапе является функциональное тестирование и структурное тестирование.

Функциональное тестирование может быть рассмотрено на примере прецедента «Список предоставляемых услуг». В данном прецеденте для успешного добавления новой позиции необходимо ввести корректные данные в поля ввода.

В самом названии могут использоваться буквы русского алфавита, цифры, символы: точка, тире, кавычки, пробел. Количество символов должно находиться в диапазоне от 5 до 150.

Реакция программы на пользовательский ввод:

- программа должна сообщать о невозможном создании услуги, если имя содержит некорректные символы;
- программа должна сообщать о неверной длине названия услуги;
- программа должна сообщать о том, что такая услуга уже существует;
- программа должна сообщать о том, что используется неверная раскладка;
- программа должна сообщить об успешном создании новой услуги, если все введенные пользователем данные оказались допустимыми.

На рисунке 25 представлены классы эквивалентности для длины названия медицинской услуги m.

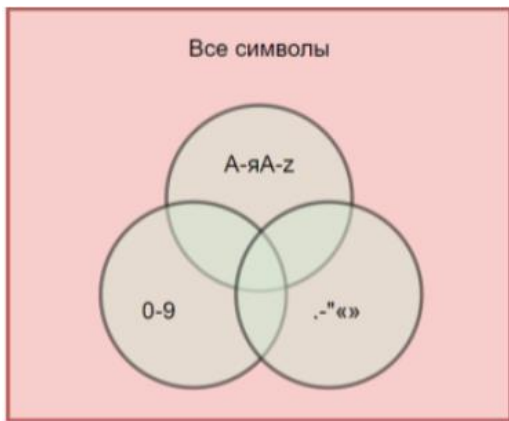


Рисунок 25 – Классы эквивалентности для названия медицинской услуги

В таблице 11 представлены классы эквивалентов для названия медицинских услуг.

Таблица 11 – Классы эквивалентов для «Список предоставляемых услуг»

№	Входные данные	Правильность класса
1	0..1	+
2	a..я, A..Я	+
3	a..z, A..Z	-
4	ø	-
5	*#, .. %	-
6	“” . -	+

В таблице 12 представлены определения границ для списка предоставляемых услуг.

Таблица 12 – Определение границ значений для поля «Список предоставляемых услуг»

Граница	Значение	Диапазон
Физическая	Строка не может быть отрицательной длины	$(0, \infty)$
Логическая	Длина названия должна быть больше 0	$(0, \infty)$
Технологическая	Существует ограничение в 65535 символов	$[0, 65535]$
Произвольная	Значение поля должно быть от 5 до 150 символов	$[5, 150]$

В таблице 13 представлены тесты по определенным границам значений и классам эквивалентности.

Таблица 13 – тесты по определенным границам значений и классам эквивалентности

№	Входные данные	Предполагаемый результат	Фактический результат
1	Рентген	Данные успешно добавлены в базу	Данные успешно добавлены в базу
2	УЗИ (1-го органа)	Данные успешно добавлены в базу	Данные успешно добавлены в базу

Продолжение таблицы 13

№	Входные данные	Предполагаемый результат	Фактический результат
3	Rentgen	Данные не могут быть добавлены в базу	Данные не могут быть добавлены в базу
4	ø	Данные не могут быть добавлены в базу	Данные не могут быть добавлены в базу
5	п	Данные не могут быть добавлены в базу	Данные не могут быть добавлены в базу
6	реер	Данные не могут быть добавлены в базу	Данные не могут быть добавлены в базу
7	Кварц	Данные успешно добавлены в базу	Данные успешно добавлены в базу
8	Клизма	Данные успешно добавлены в базу	Данные успешно добавлены в базу
9	Вентрикулоперитонеостомия	Данные успешно добавлены в базу	Данные успешно добавлены в базу
10	Визуальное исследование при патологии полости рта	Данные успешно добавлены в базу	Данные успешно добавлены в базу
11	Сбор анамнеза, жалоб при патологии костной системы	Данные успешно добавлены в базу	Данные успешно добавлены в базу
12	Сбор анамнеза и жалоб при патологии костной системы	Данные успешно добавлены в базу	Данные успешно добавлены в базу
13	Пальпация при термических, химических и электрических ожогах	Данные успешно добавлены в базу	Данные успешно добавлены в базу

Метод причинно-следственных связей – функциональная диаграмма представлена на рисунке 26.

Функциональными называют диаграммы, в первую очередь, отражающие взаимосвязи функций разрабатываемого программного обеспечения.

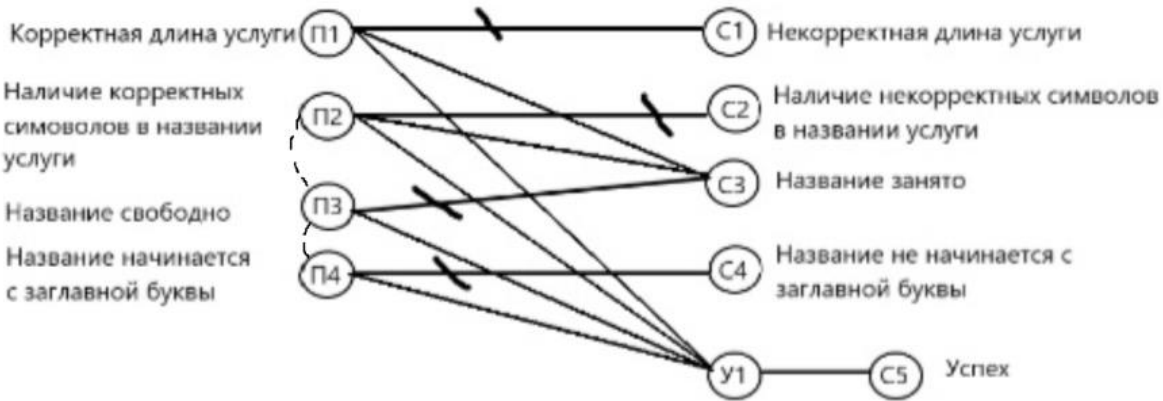


Рисунок 26 – Функциональная диаграмма для метода причинно-следственных связей

В таблице 14 представлены решения для функциональной диаграммы.

Таблица 14 – Таблица решений

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
П1	0	1	1	1	1	0	0	0	0	0	0	1	1	1	0	1
П2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
П3	0	0	0	1	1	1	0	1	1	0	0	0	1	0	1	1
П4	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1
С1	1	0	0	0	0	1	1	1	1	1	1	0	0	0	1	0
С2	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
С3	1	1	1	0	0	0	1	0	0	1	1	1	0	1	0	0
С4	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	0
С5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

На основании таблицы решений были созданы тесты, показанные в таблице 15.

Таблица 15 – Тесты по функциональной диаграмме

№	Входные данные	Предполагаемые выходные данные	Результат
1	+	Ошибка ввода	Данные не добавлены в базу
2	рентген+++	Ошибка ввода	Данные не добавлены в базу

Продолжение таблицы 15

№	Входные данные	Предполагаемые выходные данные	Результат
3	рентген	Ошибка ввода	Данные не добавлены в базу
4	рентген	Ошибка ввода	Данные не добавлены в базу
5	рен+тген	Ошибка ввода	Данные не добавлены в базу
6	р+	Ошибка ввода	Данные не добавлены в базу
7	ре	Ошибка ввода	Данные не добавлены в базу
8	Ре	Ошибка ввода	Данные не добавлены в базу
9	Рен	Ошибка ввода	Данные не добавлены в базу
10	Р	Ошибка ввода	Данные не добавлены в базу
11	Ре	Ошибка ввода	Данные не добавлены в базу
12	Рентген	Ошибка ввода	Данные не добавлены в базу
13	Р++++	Ошибка ввода	Данные не добавлены в базу
14	Р++++++	Ошибка ввода	Данные не добавлены в базу
15	рен	Ошибка ввода	Данные не добавлены в базу
16	Рентген	Данные введены	Данные добавлены в базу

Структурное тестирование может быть проведено на примере прецедента диаграммы вариантов использования «Создание новой медицинской услуги». Диаграмма деятельности представлена на рисунке 27.

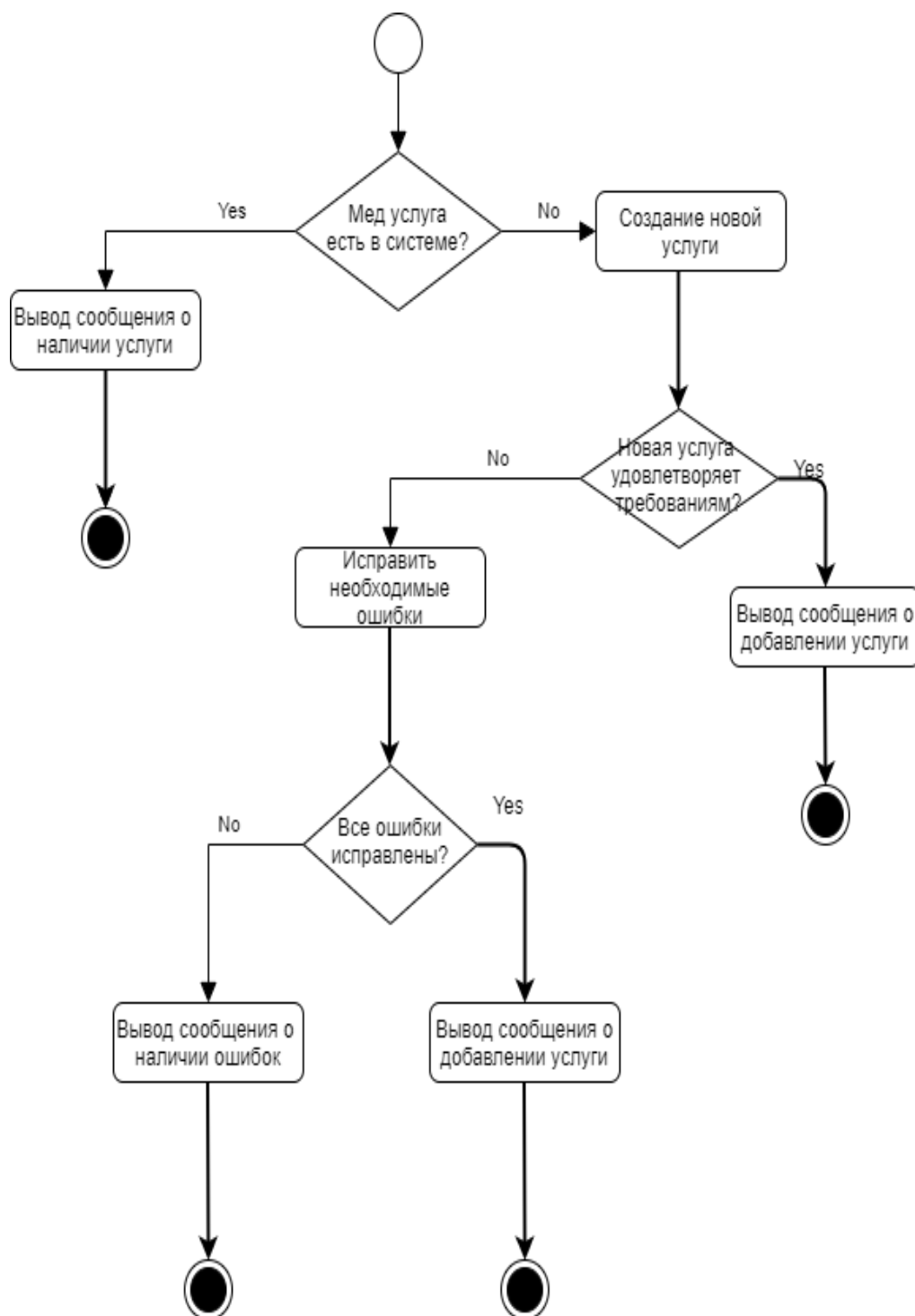


Рисунок 27 – Диаграмма деятельности прецедента «Создание новой медицинской услуги»

Граф переходов для диаграммы деятельности прецедента «Создание новой медицинской услуги» представлен на рисунке 28.

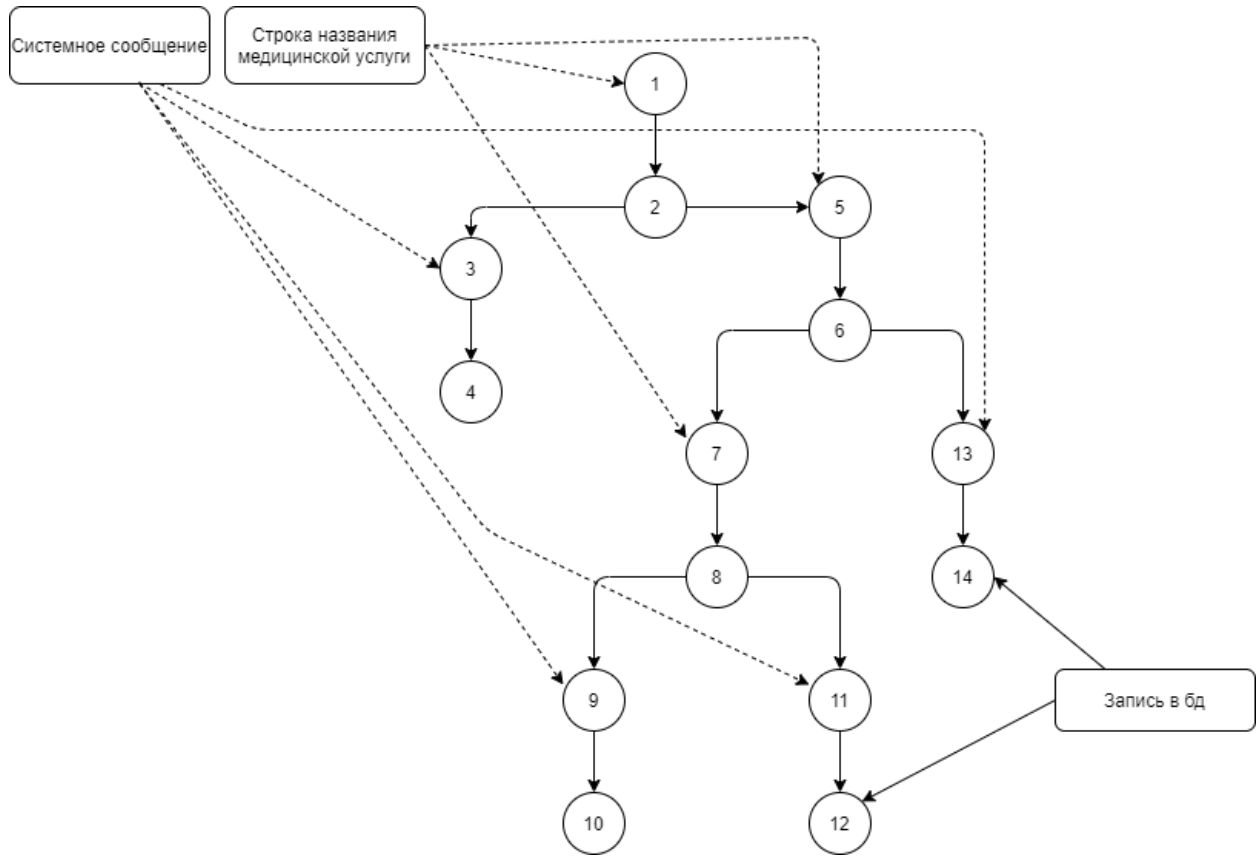


Рисунок 28 – Граф переходов для диаграммы деятельности прецедента «Создание новой медицинской услуги»

Критерии покрытия операторов подразумевает выполнение каждого оператора программы, по крайней мере, один раз.

Для выполнения каждого оператора были составлены тесты. Результаты приведены в таблице 16.

Таблица 16 – Результат тестирования методом покрытия операторов.

Операторы	Входные данные	Предполагаемый результат	Фактический результат
1, 2, 3, 4	Рентген	Имя не может быть добавлено в базу повторно	Имя не может быть добавлено в базу повторно
1, 2, 5, 6, 7, 8, 9, 10	*Узи (1-го органа)	Имя не может быть добавлено в базу	Имя не может быть добавлено в базу

Продолжение таблицы 16

Операторы	Входные данные	Предполагаемый результат	Фактический результат
1, 2, 5, 6, 7, 8, 11, 12	Узи (1-го органа)	Имя добавлено в базу	Имя добавлено в базу
1, 2, 5, 6, 13, 14	Клизма	Имя добавлено в базу	Имя добавлено в базу

Суть метода покрытия условий заключается в том, чтобы составить число тестов, достаточное для того, чтобы все возможные результаты каждого условия в решении выполнялись, по крайней мере, один раз. Также должно соблюдаться условие выполнения каждого оператора, по крайней мере, один раз.

Для выполнения каждого условия были составлены тесты. Результаты приведены в таблице 17.

Таблица 17 – Результат тестирования методом покрытия условий.

Условия	Входные данные	Предполагаемый результат	Результат тестирования
2 - да	Название мед услуги введено, но уже есть в базе	Название не добавлено повторно	Успешно
2 - нет, 6 - да	Узи (1-го органа)	Название добавлено	Успешно
2 - нет, 6 - нет, 7 - да	Узи (1-го органа)	Название добавлено	Успешно
2 - нет, 6 - нет, 8 - нет	*Узи (1-го органа)	Название не может быть добавлено	Успешно

Для выполнения всех маршрутов были составлены тесты. Результаты приведены в таблице 18.

Таблица 18– Результат тестирования методом покрытия маршрутов.

Маршрут	Тест	Фактический результат	Результат тестирования
1, 2, 3, 4	Рентген	Имя не может быть добавлено в базу повторно	Имя не может быть добавлено в базу повторно
1, 2, 5, 6, 7, 8, 9, 10	*Узи (1-го органа)	Имя не может быть добавлено в базу	Имя не может быть добавлено в базу
1, 2, 5, 6, 7, 8, 11, 12	Узи (1-го органа)	Имя добавлено в базу	Имя добавлено в базу
1, 2, 5, 6, 13, 14	Клизма	Имя добавлено в базу	Имя добавлено в базу

Для проверки запросов на работоспособность было использовано приложение Postman.

Postman – это бесплатная программа, которая предназначена для создания и тестирования работы различных API или просто отправки POST и GET запросов на сервер.

На рисунке 29 представлена работоспособность запроса «post».

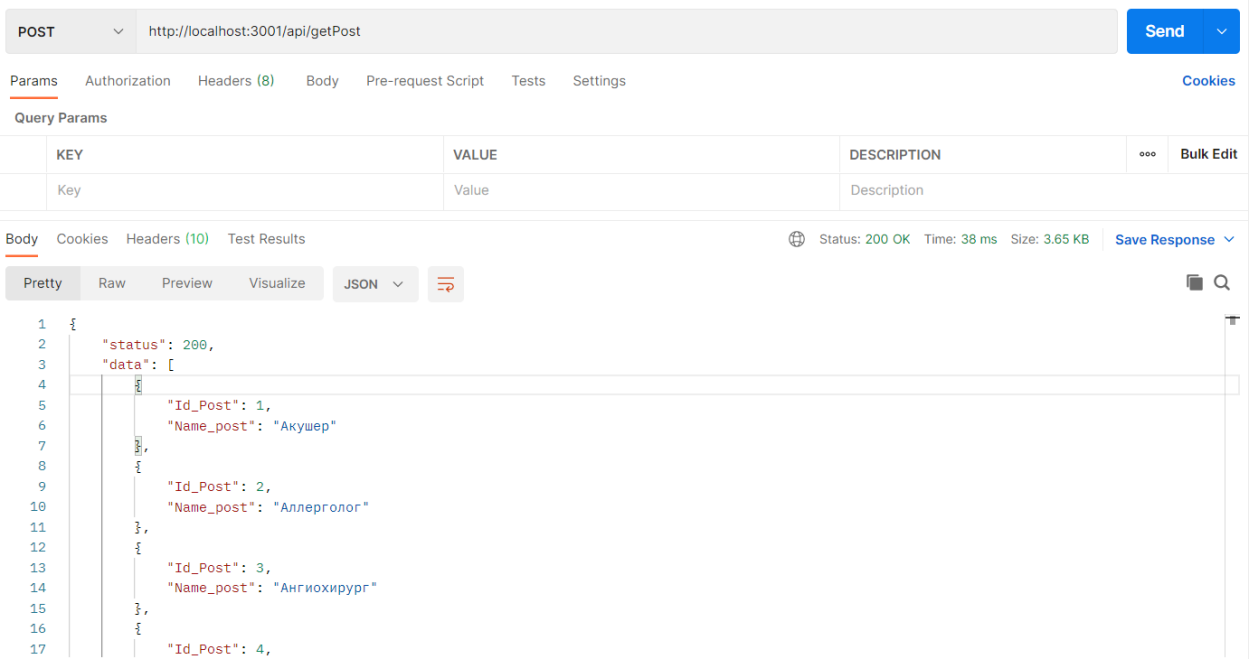


Рисунок 29 – Работоспособность запроса «post»

На рисунке 30 представлена работоспособность запроса «MedicalFacility».

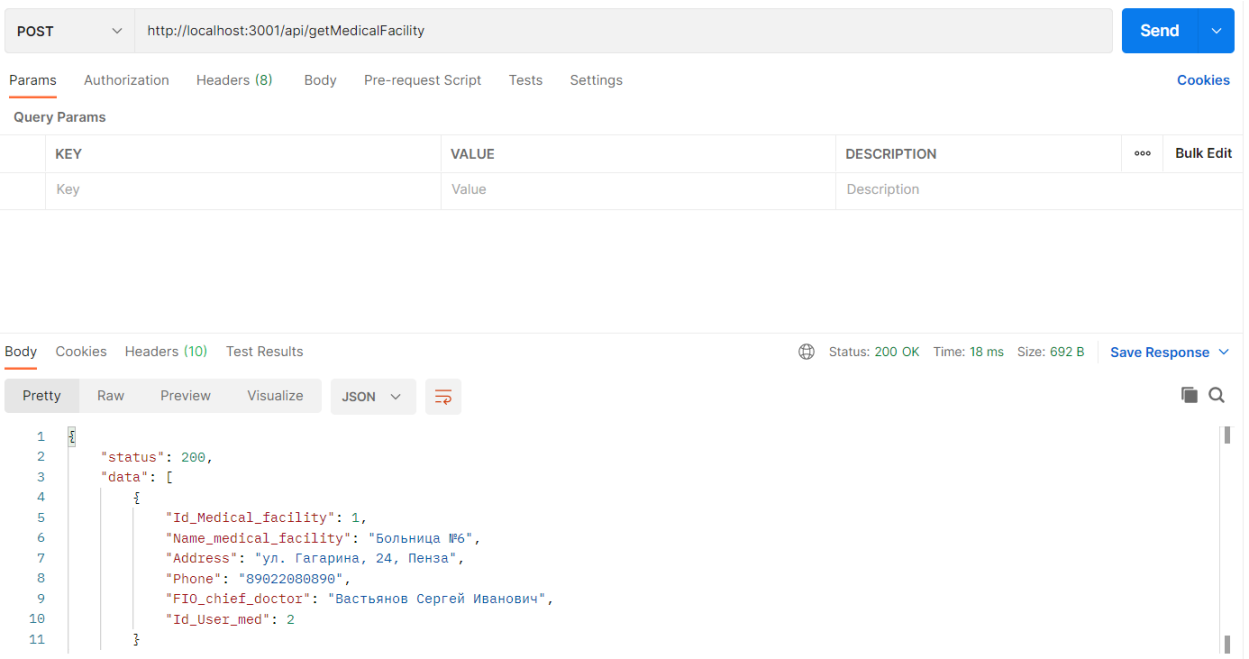


Рисунок 30 – Работоспособность запроса «MedicalFacility»

Как видно из результатов, все тесты были пройдены успешно.

Заключение

В процессе выполнения выпускной квалификационной работы был выполнен анализ предметной области, выполнена постановка задачи на разработку, был проведен сравнительный анализ с существующими системами-аналогами, выполнен анализ требований на разработку, было разработано информационное и программное обеспечение, выполнен контроль качества программного обеспечения, а также было проведено тестирование программного обеспечения. Тестирование программного обеспечения включало в себя функциональное и структурное тестирование.

В результате выполнения выпускной квалификационной работы были реализованы следующие функции web-портала для поиска и заказа медицинских услуг: регистрация и вход в систему пользователей, запись на получение медицинской услуги, просмотр медицинских учреждений города, просмотр списка врачей, просмотр медицинских услуг, а также их стоимости, возможна регистрация медицинских учреждений, возможен просмотр аналитики.

Выпускная квалификационная работа была оформлена по учебно-методическому пособию «Выпускная квалификационная работа: стандарты и программные технологии» [22].

Список использованных источников

1. Андерсен, Б., Бизнес-процессы. Инструменты совершенствования. / Б. Андерсен. – Санкт-Петербург, Издательство: РИА «Стандарты и качество», 2003.
2. Зайцева, Е. В., Делопроизводство и документооборот в системе государственного и муниципального управления. / Е. В. Зайцева, Н. В. Гончарова – Екатеринбург, Издательство «Уральский университет», 2017.
3. Нормативно-правовые документы медицинской организации [Интернет-ресурс]: учебно-практическое пособие. Режим доступа: <https://www.ligamedprava.ru/publikatsii/normativnopravovye-dokumenty-meditsinskoy-organizatsii/> (дата обращения: 25.04.2021)
4. Требования к официальному сайту медицинской организации [Интернет-ресурс]. Режим доступа: <https://fireseo.ru/articles/trebovaniya-k-sajtu-medicinskoj-organizacii/> (дата обращения: 25.04.2021)
5. Основы функционально-стоимостного анализа [Интернет-ресурс]: учебно-практическое пособие. Режим доступа: <https://yandex.ru/search/?text=Основы%20функционально-стоимостного%20анализа&clid=2270455&banerid=0500000134%3A5c1d117f67a8eb0018c4df2f&win=364&&lr=18> (дата обращения: 25.04.2021)
6. Методика функционально-стоимостного анализа [Интернет-ресурс]: учебно-практическое пособие. Режим доступа: <https://intuit.ru/studies/courses/3636/878/lecture/32286?page=2> (дата обращения: 25.04.2021)
7. Бояркин, Г. Н., Моделирование бизнес-процессов/ Г. Н. Бояркин, К. В. Кравченко. – Омск, Издательство: «ОмГТУ», 2019.
8. 4 типа архитектуры программного обеспечения [Интернет-ресурс]: учебно-практическое пособие. Режим доступа: <https://nuancesprog.ru/p/12019/> (дата обращения: 01.06.2021)
9. Кратко о типах архитектур программного обеспечения и переходе на микросервисы [Интернет-ресурс]. Режим доступа: <https://spark.ru/startup/1cloud/blog/42506/kratko-o-tipah-arhitektur->

programmno-obespecheniya-i-perehode-na-mikroservisi (дата обращения: 01.06.2021)

10. Обоснование выбора СУБД [Интернет-ресурс]: учебно-практическое пособие. Режим доступа: https://studbooks.net/2249762/informatika/obosnovanie_vybora_subd (дата обращения: 01.06.2021)

11. Пустобаева, А. А., Обзор современных систем управления базами данных / А. А. Пустобаева, И. А. Казакова, - Пенза, Издательство: «ПГУ», 2019 г.

12. Обоснование выбора систем управления базами данных [Интернет-ресурс]. Режим доступа: https://studbooks.net/2005578/informatika/obosnovanie_vybora_sistem_upravleniya_bazami_dannyh (дата обращения: 01.06.2021)

13. Обоснование выбора используемой СУБД [Интернет-ресурс]. Режим доступа: https://studopedia.net/6_59049_obosnovanie-vibora-ispolzuemoj-subd.html#:~:text=Выбор%20системы%20управления%20базами%20данных,основе%2C%20а%20также%20обучение%20персонала (дата обращения: 01.06.2021)

14. Роберт, У. Себеста. Основные концепции языков программирования / У. Себеста Роберт. – Пер. с англ. — 5-е изд. — М.: Вильямс, 2001.

15. Библиотека React: особенности, перспективы, ситуация на рынке труда [Интернет-ресурс]. Режим доступа: <https://ru.hexlet.io/blog/posts/biblioteka-react-review-article> (дата обращения: 01.06.2021)

16. Тиленс, Томас М. React в действии / М. Тиленс Томас – Санкт-Петербург, Издательство: Эксмо-АСТ, 2019.

17. Что такое MVC - шаблон проектирования model view controller (MVC) и PHP, Часть 1 [Интернет-ресурс]. Режим доступа: <https://www.internet->

technologies.ru/articles/shablon-proektirovaniya-mvc-i-php-chast-1.html (дата обращения: 01.06.2021)

18. Фримен, А. ASP.NET Core MVC 3 с примерами на C# для профессионалов / А. Фримен – Москва, Издательство: Вильямс, 2021.
19. Арно, Л. Проектирование веб-API / Л. Арно – М.: ДМК Пресс, 2020.
20. Белик, А. Г., Качество и надежность программных систем / А. Г. Белик, В. Н. Цыганенко – Омск, Издательство: «ОмГТУ», 2019.
21. Куликов, С. Тестирование программного обеспечения. Базовый курс / С. Куликов – Минск, Издательство: ЕРАМ Systems, 2020.
22. Гурьянов, Л. В. Выпускная квалификационная работа: стандарты и программные технологии / Л. В. Гурьянов, В. Н. Князев, И. Ю. Балашова, Е. А. Дзюба, Н. А. Попова, Д. В. Такташкин, С. В. Самуйлов – Пенза: Издательство ПГУ, 2020. – 154 с.

Приложение А

Описание диаграмм деятельности

(рекомендуемое)

Таблица А.1 – Описание диаграммы деятельности для узла «Зарегистрироваться в системе»

Узел «Зарегистрироваться в системе»
Тип: Узел действия
Описание: Перед тем, как начать работать с системой, если пользователь еще не зарегистрирован, то должен зарегистрироваться.
Входные потоки: –
Выходные потоки: «Войти в систему»

Таблица А.2 – Описание диаграммы деятельности для узла «Войти в систему»

Узел «Войти в систему»
Тип: Узел действия
Описание: Перед тем, как начать работать с системой, пользователь должен войти в систему.
Входные потоки: «Зарегистрироваться в системе»
Выходные потоки: «Создать заявку на услугу»

Таблица А.3 – Описание диаграммы деятельности для узла «Создать заявку на услугу»

Узел «Создать заявку на услугу»
Тип: Узел действия
Описание: Пользователь создает заявку на необходимую ему услугу.
Входные потоки: «Войти в систему»
Выходные потоки: «Заполнить заявку на услугу»

Таблица А.4 – Описание диаграммы деятельности для узла «Заполнить заявку на услугу»

Узел «Заполнить заявку на услугу»
Тип: Узел действия
Описание: Пользователь заполняет созданную заявку на услугу.
Входные потоки: «Создать заявку на услугу»
Выходные потоки: «Выбрать дату и время для получения услуги»

Таблица А.5 – Описание диаграммы деятельности для узла «Выбрать дату и время для получения услуги»

Узел «Выбрать дату и время для получения услуги»
Тип: Узел действия
Описание: Пользователь выбирает удобное ему время и дату для получения услуги.
Входные потоки: «Заполнить заявку на услугу»
Выходные потоки: «Проверить заполненные пользователем данные»

Таблица А.6 – Описание диаграммы деятельности для узла «Проверить заполненные пользователем данные»

Узел «Проверить заполненные пользователем данные»
Тип: Узел действия
Описание: Система проверяет данные, введенные пользователем в заявке на услугу.
Входные потоки: «Выбрать дату и время для получения услуги»
Выходные потоки: «Проверить введенные пользователем дату и время»

Таблица А.7 – Описание диаграммы деятельности для узла «Проверить введенные пользователем дату и время»

Узел «Проверить введенные пользователем дату и время»
Тип: Узел действия
Описание: Система проверяет дату и время, выбранное пользователем.
Входные потоки: «Проверить заполненные пользователем данные»
Выходные потоки: «Сообщить, что дата/время заняты»

Таблица А.8 – Описание диаграммы деятельности для узла «Сообщить, что дата/время заняты»

Узел «Сообщить, что дата/время заняты»
Тип: Узел действия
Описание: Система сообщает пользователю о том, что выбранное им время или дата уже заняты.
Входные потоки: «Проверить введенные пользователем дату и время»
Выходные потоки: «Предложить пользователю свободную дату/время»

Таблица А.9 – Описание диаграммы деятельности для узла «Предложить пользователю свободную дату/время»

Узел «Предложить пользователю свободную дату/время»
Тип: Узел действия
Описание: Система предлагает пользователю свободную дату и время из имеющихся.
Входные потоки: «Сообщить, что дата/время заняты»
Выходные потоки: «Выбрать дату и время из предложенных системой»

Таблица А.10 – Описание диаграммы деятельности для узла «Выбрать дату и время из предложенных системой»

Узел «Выбрать дату и время из предложенных системой»
Тип: Узел действия
Описание: Пользователь выбирает свободную дату и время, предложенные системой.
Входные потоки: «Предложить пользователю свободную дату/время»
Выходные потоки: «Уведомить пользователя об успешно созданной заявке»

Таблица А.11 – Описание диаграммы деятельности для узла «Уведомить пользователя об успешно созданной заявке»

Узел «Уведомить пользователя об успешно созданной заявке»
Тип: Узел действия
Описание: Система уведомляет клиента, что заявка успешно создана.
Входные потоки: «Выбрать дату и время из предложенных системой»
Выходные потоки: –

Таблица А.12 – Описание диаграммы деятельности для узла «Зарегистрироваться в системе»

Узел «Зарегистрироваться в системе»
Тип: Узел действия
Описание: Перед тем, как начать работать с системой, если пользователь еще не зарегистрирован, то должен зарегистрироваться.
Входные потоки: –
Выходные потоки: «Войти в систему»

Таблица А.13 – Описание диаграммы деятельности для узла «Войти в систему»

Узел «Войти в систему»
Тип: Узел действия
Описание: Перед тем, как начать работать с системой, пользователь должен войти в систему.
Входные потоки: «Зарегистрироваться в системе»
Выходные потоки: «Выбрать автоматический подбор специалиста по введенным симптомам»

Таблица А.14 – Описание диаграммы деятельности для узла «Выбрать автоматический подбор специалиста по введенным симптомам»

Узел «Выбрать автоматический подбор специалиста по введенным симптомам»
Тип: Узел действия
Описание: Пользователь должен выбрать услугу автоматического подбора специалиста по симптомам.
Входные потоки: «Войти в систему»
Выходные потоки: «Ввести все имеющиеся симптомы»

Таблица А.15 – Описание диаграммы деятельности для узла «Ввести все имеющиеся симптомы»

Узел «Ввести все имеющиеся симптомы»
Тип: Узел действия
Описание: Пользователь должен ввести все симптомы, которые у него присутствуют для дальнейшего подбора специалиста.
Входные потоки: «Выбрать автоматический подбор специалиста по введенным симптомам»
Выходные потоки: «Осуществить подбор специалистов по симптомам»

Таблица А.16 – Описание диаграммы деятельности для узла «Осуществить подбор специалистов по симптомам»

Узел «Осуществить подбор специалистов по симптомам»
Тип: Узел действия
Описание: По введенным пользователем данным система осуществляет подбор специалистов.
Входные потоки: «Ввести все имеющиеся симптомы»
Выходные потоки: «Вывести список врачей», «Вывести сообщение о том, что данных мало»

Таблица А.17 – Описание диаграммы деятельности для узла «Вывести список врачей»

Узел «Вывести список врачей»
Тип: Узел действия
Описание: Система выводит список подходящих по симптому пользователя врачей
Входные потоки: «Осуществить подбор специалистов по симптомам»
Выходные потоки: –

Таблица А.18 – Описание диаграммы деятельности для узла «Вывести сообщение о том, что данных мало»

Узел «Вывести сообщение о том, что данных мало»
Тип: Узел действия
Описание: Система выводит сообщение о том, что пользователь ввёл мало данных и их не хватает для подбора конкретного специалиста.
Входные потоки: «Осуществить подбор специалистов по симптомам»
Выходные потоки: –

Таблица А.19 – Описание диаграммы деятельности для узла «Зарегистрироваться в системе»

Узел «Зарегистрироваться в системе»
Тип: Узел действия
Описание: Перед тем, как начать работать с системой, если пользователь еще не зарегистрирован, то должен зарегистрироваться.
Входные потоки: –
Выходные потоки: «Войти в систему»

Таблица А.20 – Описание диаграммы деятельности для узла «Войти в систему»

Узел «Войти в систему»
Тип: Узел действия
Описание: Перед тем, как начать работать с системой, пользователь должен войти в систему.
Входные потоки: «Зарегистрироваться в системе»
Выходные потоки: «Выбрать автоматический подбор ближайшего медицинского учреждения»

Таблица А.21 – Описание диаграммы деятельности для узла «Выбрать автоматический подбор ближайшего медицинского учреждения»

Узел «Выбрать автоматический подбор ближайшего медицинского учреждения»
Тип: Узел действия
Описание: Пользователь должен выбрать услугу автоматического подбора ближайшего медицинского учреждения.
Входные потоки: «Войти в систему»
Выходные потоки: «Ввести адрес проживания»

Таблица А.22 – Описание диаграммы деятельности для узла «Ввести адрес проживания»

Узел «Ввести адрес проживания»
Тип: Узел действия
Описание: Пользователь должен ввести адрес проживания для подбора ближайшего медицинского учреждения
Входные потоки: «Выбрать автоматический подбор ближайшего медицинского учреждения»
Выходные потоки: «Ввести специальность врача»

Таблица А.23 – Описание диаграммы деятельности для узла «Ввести специальность врача»

Узел «Ввести специальность врача»
Тип: Узел действия
Описание: Пользователь должен ввести специальность врача, чтобы система могла подобрать необходимое ближайшее медицинское учреждение с таким специалистом
Входные потоки: «Ввести адрес проживания»
Выходные потоки: «Вывести ближайшие учреждения»

Таблица А.24 – Описание диаграммы деятельности для узла «Вывести ближайшие учреждения»

Узел «Вывести ближайшие учреждения»
Тип: Узел действия
Описание: Система выводит ближайшие учреждения.
Входные потоки: «Ввести специальность врача»
Выходные потоки: «Вывести расписание врачей нужной специальности»

Таблица А.25 – Описание диаграммы деятельности для узла «Вывести расписание врачей нужной специальности»

Узел «Вывести расписание врачей нужной специальности»
Тип: Узел действия
Описание: Система выводит пользователю расписание необходимого специалиста
Входные потоки: «Вывести ближайшие учреждения»
Выходные потоки: –

Приложение Б
Описание концептуальной модели
(рекомендуемое)

Таблица Б.1 – Описание диаграммы классов («Client»)

Имя класса	Client
Назначение	Описывает клиента, который записывается на приём в медучреждение
Атрибуты	Policy_number : Char — номер полиса клиента Date_of_birth : Date — дата рождения клиента SNILS : Char — номер снилса клиента
Методы	Make_an_appointment(id_Client,id_Employee,id_Employee_schedule) — запись на прием клиента To_evaluate_the_services_provided(id_Feedback) — оценивание клиентом оказанных ему услуг
Связь	Feedback — ассоциация. Ассоциация показывает, что объекты класса Client связаны с объектами класса Feedback таким образом, что можно перемещаться от объектов одного класса к другому. Make_an_appointment_with_a_doctor — агрегация. Агрегация показывает, что объекты класса Make_an_appointment_with_a_doctor являются целым, а Client его частью. Registration_for_a_laboratory_examination — агрегация. Агрегация показывает, что объекты класса Registration_for_a_laboratory_examination являются целым, а Client его частью. User – обобщение. Обобщение показывает отношение между более общей сущностью User и более специальной сущностью Client.

Таблица Б.2 – Описание диаграммы классов («User»)

Имя класса	User
Назначение	Описывает пользователя, который регистрируется в системе
Атрибуты	Surname : String - Фамилия Name : String - Имя Patronymic : String - Отчество Passport_data : Char — Паспортные данные Registration_date : Date — Дата регистрации Login : String — Логин Password : String — Пароль
Методы	Authorization() - авторизация пользователя в системе
Связь	Client - обобщение. Обобщение показывает отношение между более общей сущностью User и более специальной сущностью Client. Administrator - обобщение. Обобщение показывает отношение между более общей сущностью User и более специальной сущностью Administrator.

Таблица Б.3 – Описание диаграммы классов («Administrator»)

Имя класса	Administrator
Назначение	Администратор, который регистрируется и регистрирует медучреждение и загружает все необходимые данные в систему о нем
Атрибуты	-
Методы	Uploading_information_about_a_medical_facility — загрузка сведений о медучреждении Uploading_employee_information — загрузка сведений о сотрудниках The_download_of_employee_scheduling — загрузка сведений о расписании сотрудников
Связь	User – обобщение. Обобщение показывает отношение между более общей сущностью User и более специальной сущностью Administrator. Medical_facility - ассоциация.

Таблица Б.4 – Описание диаграммы классов («System_administrator»)

Имя класса	System_administrator
Назначение	Администратор, который регистрируется и регистрирует медучреждение и загружает все необходимые данные в систему о нем
Атрибуты	-
Методы	Registration_of_medical_facility — регистрация медучреждения в системе
Связь	User – обобщение. Обобщение показывает отношение между более общей сущностью User и более специальной сущностью Administrator. Medical_facility - ассоциация. Ассоциация показывает, что объекты класса Administrator связаны с объектами класса Medical_facility таким образом, что можно перемещаться от объектов одного класса к другому.

Таблица Б.5 – Описание диаграммы классов («Employee»)

Имя класса	Employee
Назначение	Сотрудник медучреждения
Атрибуты	Surname : String - Фамилия Name : String - Имя Patronymic : String – Отчество Telephone : Char - номер телефона
Методы	-
Связь	Medical_facility - ассоциация. Ассоциация показывает, что объекты класса Employee связаны с объектами класса Medical_facility таким образом, что можно перемещаться от объектов одного класса к другому. Make_an_appointment_with_a_doctor - ассоциация. Registration_for_a_laboratory_examination - ассоциация. Ассоциация показывает, что объекты класса Registration_for_a_laboratory_examination связаны с объектами класса Employee таким образом, что можно перемещаться от объектов одного класса к другому. Employee_schedule - ассоциация. Ассоциация показывает, что объекты класса Employee_schedule связаны с объектами класса Employee таким образом, что можно перемещаться от объектов одного класса к другому.

Таблица Б.6 – Описание диаграммы классов («Feedback»)

Имя класса	Feedback
Назначение	Отзыв
Атрибуты	Evaluation : Byte — оценка Description : String — описание
Методы	—
Связь	Client - ассоциация. Ассоциация показывает, что объекты класса Client связаны с объектами класса Feedback таким образом, что можно перемещаться от объектов одного класса к другому.

Таблица Б.7 – Описание диаграммы классов («Medical_facility»)

Имя класса	Medical_facility
Назначение	Медицинское учреждение
Атрибуты	Title : String — Название медучреждения Address : String — адрес медучреждения Telephone : Char — телефон медучреждения FIO_chief_doctor : String — ФИО главного врача Type_of_medical_facility : Enum- тип медучреждения
Методы	Get_information_about_a_medical_facility() - получать информацию о медучреждении
Связь	Administrator - ассоциация. Ассоциация показывает, что объекты класса Administrator связаны с объектами класса Medical_facility таким образом, что можно перемещаться от объектов одного класса к другому. Employee - ассоциация. Ассоциация показывает, что объекты класса Employee связаны с объектами класса Medical_facility таким образом, что можно перемещаться от объектов одного класса к другому. Make_an_appointment_with_a_doctor - ассоциация. Ассоциация показывает, что объекты класса Make_an_appointment_with_a_doctor связаны с объектами класса Medical_facility таким образом, что можно перемещаться от объектов одного класса к другому. Registration_for_a_laboratory_examination - ассоциация. Ассоциация показывает, что объекты класса Registration_for_a_laboratory_examination связаны с объектами класса Medical_facility таким образом, что можно перемещаться от объектов одного класса к другому. Analyzes - ассоциация. Ассоциация показывает, что объекты класса Analyzes связаны с объектами класса Medical_facility таким образом, что можно перемещаться от объектов одного класса к другому.

Таблица Б.8 – Описание диаграммы классов («Make_an_appointment_with_a_doctor»)

Имя класса	Make_an_appointment_with_a_doctor
Назначение	Запись на прием к врачу
Атрибуты	Date : Date — Дата приема Time : Time — Время приема
Методы	-
Связь	<p>Client — ассоциация. Ассоциация показывает, что объекты класса Make_an_appointment_with_a_doctor связаны с объектами класса Client таким образом, что можно перемещаться от объектов одного класса к другому.</p> <p>Medical_facility - ассоциация. Ассоциация показывает, что объекты класса Make_an_appointment_with_a_doctor связаны с объектами класса Medical_facility таким образом, что можно перемещаться от объектов одного класса к другому.</p> <p>Employee - ассоциация. Ассоциация показывает, что объекты класса Make_an_appointment_with_a_doctor связаны с объектами класса Employee таким образом, что можно перемещаться от объектов одного класса к другому.</p>

Таблица Б.9 – Описание диаграммы классов («Registration_for_a_laboratory_examination»)

Имя класса	Registration_for_a_laboratory_examination
Назначение	Запись на лабораторное обследование
Атрибуты	Date : Date — Дата приема Time : Time — Время приема
Методы	-
Связь	<p>Client — ассоциация. Ассоциация показывает, что объекты класса Registration_for_a_laboratory_examination связаны с объектами класса Client таким образом, что можно перемещаться от объектов одного класса к другому.</p> <p>Medical_facility - ассоциация. Ассоциация показывает, что объекты класса Registration_for_a_laboratory_examination связаны с объектами класса Medical_facility таким образом, что можно перемещаться от объектов одного класса к другому.</p> <p>Employee - ассоциация. Ассоциация показывает, что объекты класса Registration_for_a_laboratory_examination связаны с объектами класса Employee таким образом, что можно перемещаться от объектов одного класса к другому.</p> <p>Analyzes - ассоциация. Ассоциация показывает, что объекты класса Registration_for_a_laboratory_examination связаны с объектами класса Analyzes таким образом, что можно перемещаться от объектов одного класса к другому.</p>

Таблица Б.10 – Описание диаграммы классов («Analyzes»)

Имя класса	Analyzes
Назначение	Анализы
Атрибуты	Type_of_analysis : String — тип анализа Name_of_the_analysis : String — название анализа Cost_of_analysis : Char — стоимость анализа Term_of_execution : String — срок исполнения
Методы	-
Связь	Registration_for_a_laboratory_examination - ассоциация. Ассоциация показывает, что объекты класса Registration_for_a_laboratory_examination связаны с объектами класса Analyzes таким образом, что можно перемещаться от объектов одного класса к другому. Medical_facility - ассоциация. Ассоциация показывает, что объекты класса Analyzes связаны с объектами класса Medical_facility таким образом, что можно перемещаться от объектов одного класса к другому.

Таблица Б.11– Описание диаграммы классов («Employee_schedule»)

Имя класса	Employee_schedule
Назначение	Расписание сотрудников
Атрибуты	Date : Date - Дата Reception_start_time : Time — Время начала приема The_end_time_of_the_reception : Time - Время окончания приема
Методы	-
Связь	Employee - ассоциация. Ассоциация показывает, что объекты класса Employee_schedule связаны с объектами класса Employee таким образом, что можно перемещаться от объектов одного класса к другому.

Приложение В
Описание сущностей базы данных
(рекомендуемое)

Таблица В.1 – Сущность «user_med»

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения NULL	Роль
Id_User_med	Код пользователя	целый		Нет	РК
Surname	Фамилия пользователя	символьный	50	Нет	
Name	Имя пользователя	символьный	50	Нет	
Patronymic	Отчество пользователя	символьный	50	Да	
Passport_data	Паспортные данные	символьный	50	Нет	
Registration_data	Дата регистрации	Дата	8	Нет	
Role	Роль	символьный	50	Нет	
Login	Логин	символьный	50	Нет	
Password	Пароль	символьный	50	Нет	

Таблица В.2 – Сущность «client»

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения NULL	Роль
Id_Client	Код клиента	целый		Нет	РК
Policy_number	Номер полиса	целый	16	Нет	
Date_of_birt	Дата рождения	дата	8	Нет	
SNILS	Номер снилса	целый	11	Нет	
Id_User_med	Код пользователя	целый		Нет	FK

Таблица В.3 – Сущность «medical_facility»

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения NULL	Роль
Id_Medical_facility	Код медицинского учреждения	целый		Нет	РК
Name_medical_facility	Наименование мед. учреждения	символьный	50	Нет	
Address	Адрес	символьный	100	Нет	
Phone	Контактный номер	символьный	20	Нет	
FIO_chief_doctor	ФИО главного врача	символьный	100	Нет	
Id_User_med	Код пользователя	целый		Нет	FK

Таблица В.4 – Сущность «service»

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения NULL	Роль
Id_Service	Код медицинской услуги	целый		Нет	РК
Name_service	Наименование мед. услуги	символьный	150	Нет	
Cost	Стоимость	символьный	15	Да	
Id_Medical_facility	Код мед. учр.	целый		Нет	FK

Таблица В.5 – Сущность «cabinet»

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения NULL	Роль
Id_Cabinet	Код кабинета	целый		Нет	РК
Name_cabinet	Наименование кабинета	символьный	30	Нет	
Id_Medical_facility	Код мед. учр.	целый		Нет	FK

Таблица В.6 – Сущность «post»

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения NULL	Роль
Id_Post	Код должности	целый		Нет	РК
Name_post	Наименование должности	символьный	150	Нет	

Таблица В.7 – Сущность «day_of_the_week»

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения NULL	Роль
Id_Day_of_the_week	Код дня недели	целый		Нет	РК
Name_day_of_the_week	Наименование дня недели	символьный	20	Нет	

Таблица В.8 – Сущность «employee_schedule»

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения NULL	Роль
Id_Employee_schedule	Код расписания	целый		Нет	РК
Id_Day_of the_week	Код дня недели	целый		Нет	FK
Start_time	Начало приема	дата/время		Нет	

Таблица В.9 – Сущность «feedback»

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения NULL	Роль
Id_Feedback	Код отзыва	целый		Нет	РК
Evaluation	Оценка	целый		Нет	
Description	Отзыв	символьный	30	Нет	
Id_Client	Код клиента	целый		Нет	FK
Id_Record	Код записи	целый		Нет	FK

Таблица В.10 – Сущность «record»

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения NULL	Роль
Id_Record	Код отзыва	целый		Нет	РК
Date	Дата записи на мед. учлугу	дата		Нет	
Time	Дата записи на мед. учлугу	время		Нет	
Id_Medical_worker	Код медицинского работника	целый		Нет	FK
Id_Client	Код клиента	целый		Нет	FK

Таблица В.11 - Сущность «medical_worker»

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения NULL	Роль
Id_medical_worker	Код медицинского работника	целый		Нет	РК
Surname	Фамилия	символьный	30	Нет	
Name	Имя	символьный	30	Нет	
Patronymic	Отчество	символьный	30	Да	
Phone	Номер	символьный	20	Нет	
Id_Post	Код должности	целый		Нет	FK
Id_Cabinet	Код кабинета	целый		Нет	FK
Id_Medical_facility	Код медицинского учреждения	целый		Нет	FK
Id_Employee_schedule	Код расписания	целый		Нет	FK

Приложение Г

КОД СОЗДАНИЯ БАЗЫ ДАННЫХ

(рекомендуемое)

```
CREATE TABLE User_med
```

```
(
    Id_User_med    integer NOT NULL PRIMARY KEY AUTO_INCREMENT,
    Surname        varchar(50) NOT NULL ,
    Name           varchar(50) NOT NULL ,
    Patronymic     varchar(50) NULL ,
    Passport_data  varchar(50) NOT NULL ,
    Registration_data    datetime NOT NULL ,
    Role           varchar(50) NOT NULL ,
    Login          varchar(50) NOT NULL ,
    Password       varchar(50) NOT NULL
);
```

```
CREATE TABLE Client
```

```
(
    Id_Client      integer NOT NULL PRIMARY KEY AUTO_INCREMENT,
    Policy_number  varchar(16) NOT NULL ,
    Date_of_birt   datetime NOT NULL ,
    SNILS          varchar(11) NOT NULL,
    Id_User_med    integer,
    FOREIGN KEY (Id_User_med) REFERENCES user_med (Id_User_med)
);
```

```
CREATE TABLE Medical_facility
```

```
(
    Id_Medical_facility integer NOT NULL PRIMARY KEY AUTO_INCREMENT,
    Name_medical_facility varchar(50) NOT NULL ,
    Address             varchar(100) NOT NULL ,
    Phone               varchar(20) NOT NULL,
    FIO_chief_doctor    varchar(100) NOT NULL,
    Id_User_med         integer,
    FOREIGN KEY (Id_User_med) REFERENCES user_med (Id_User_med)
);
```

```
CREATE TABLE Day_of_the_week
```

```
(
    Id_Day_of_the_week    integer NOT NULL PRIMARY KEY AUTO_INCREMENT,
    Name_day_of_the_week  varchar(20) NOT NULL
);
```

```
CREATE TABLE Employee_schedule
```

```
(
    Id_Employee_schedule integer NOT NULL PRIMARY KEY
    AUTO_INCREMENT,
    Id_Day_of_the_week integer,
    FOREIGN KEY (Id_Day_of_the_week) REFERENCES Day_of_the_week
    (Id_Day_of_the_week),
    Start_time           time NOT NULL
);
```

```
CREATE TABLE Post
```

```
(
    Id_Post          integer NOT NULL PRIMARY KEY AUTO_INCREMENT,
    Name_post        varchar(150) NOT NULL
);
```

```
);
```

```
CREATE TABLE Cabinet
```

```
(
    Id_Cabinet          integer NOT NULL PRIMARY KEY AUTO_INCREMENT,
    Name_cabinet        varchar(30) NOT NULL ,
    Id_Medical_facility integer,
    FOREIGN KEY (Id_Medical_facility) REFERENCES Medical_facility (Id_Medical_facility)
);
```

```
CREATE TABLE Medical_worker
```

```
(
    Id_Medical_worker   integer NOT NULL PRIMARY KEY AUTO_INCREMENT,
    Surname             varchar(30) NOT NULL ,
    Name               varchar(30) NOT NULL ,
    Patronymic         varchar(30) NULL ,
    Phone              varchar(20) NOT NULL ,
    Id_Post             integer,
    FOREIGN KEY (Id_Post) REFERENCES Post (Id_Post),
    Id_Medical_facility integer,
    FOREIGN KEY (Id_Medical_facility) REFERENCES Medical_facility
(Id_Medical_facility),
    Id_Cabinet integer,
    FOREIGN KEY (Id_Cabinet) REFERENCES Cabinet (Id_Cabinet),
    Id_Employee_schedule integer,
    FOREIGN KEY (Id_Employee_schedule) REFERENCES Employee_schedule
(Id_Employee_schedule)
);
```

```
CREATE TABLE Record
```

```
(
    Id_Record   integer NOT NULL PRIMARY KEY AUTO_INCREMENT,
    Date        date NOT NULL ,
    Time        time NOT NULL ,
    Id_Client   integer,
    FOREIGN KEY (Id_Client) REFERENCES Client (Id_Client),
    Id_Medical_facility integer,
    FOREIGN KEY (Id_Medical_facility) REFERENCES Medical_facility
(Id_Medical_facility),
    Id_Medical_worker integer,
    FOREIGN KEY (Id_Medical_worker) REFERENCES Medical_worker (Id_Medical_worker)
);
```

```
CREATE TABLE Feedback
```

```
(
    Id_Feedback   integer NOT NULL PRIMARY KEY AUTO_INCREMENT,
    Evaluation     integer NOT NULL ,
    Description    varchar(30) NOT NULL ,
    Id_Client     integer,
    FOREIGN KEY (Id_Client) REFERENCES Client (Id_Client),
    Id_Record     integer,
    FOREIGN KEY (Id_Record) REFERENCES Record (Id_Record)
);
```

```
CREATE TABLE Service
(
    Id_Service integer NOT NULL PRIMARY KEY AUTO_INCREMENT,
    Name_service varchar(150) NOT NULL ,
    Cost varchar(15) NULL ,
    Id_Medical_facility integer,
    FOREIGN KEY (Id_Medical_facility) REFERENCES Medical_facility (Id_Medical_facility)
);
```

Приложение Д
КОД СОЗДАНИЯ СЕРВЕРА
(рекомендуемое)

```
app.js

var createError = require('http-errors');
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');

var api = require('./routes/api')

//var indexRouter = require('./routes/index');
//var usersRouter = require('./routes/users');
var dbConnection = require('./db');

var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');

app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.use('/api', api);
//app.use('/', indexRouter);
//app.use('/users', usersRouter);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  next(createError(404));
});
```

```

// error handler
app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render('error');
});

module.exports = app;

db.js
const mysql = require('mysql')

const sqlConfig = {
  host: "localhost",
  user: "root",
  password: 'root',
  database: 'medical_vkr'
}

const connection = mysql.createConnection(sqlConfig);
connection.connect(err => {
  if (err) throw err;
  console.log("успешно соединено с базой данных");
});

module.exports = connection;

api.js
const express = require('express');
```

```

const router = express.Router();
const db = require('./db');
const bodyParser = require("body-parser");
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken')

router.use(async (req, res, next) => {
  res.setHeader('Access-Control-Allow-Origin', "http://localhost:3000");
  res.setHeader("Access-Control-Allow-Methods", "GET, PUT, POST, DELETE");
  res.setHeader("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type,
Accept, Authorization");
  next();
});

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

  const sql = "SELECT * FROM medical_facility";

  db.query(sql, function (err, data, fields) {
    if (err) {
      next(new Error('Ошибка сервера!'));
    }

    res.json({
      status: 200,
      data: data
    });

  });

}, function (err, req, res, next) {
  res.json({
    status: 501,

```

```

        message: err.message
    })
});

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

    const sql = "SELECT * FROM post";

    db.query(sql, function (err, data, fields) {
        if (err) {
            next(new Error('Ошибка сервера!'));
        }

        res.json({
            status: 200,
            data: data
        });

    });

}, function (err, req, res, next) {
    res.json({
        status: 501,
        message: err.message
    })
});

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

    const sql = "SELECT * FROM medical_worker";

    db.query(sql, function (err, data, fields) {
        if (err) {

```

```

        next(new Error('Ошибка сервера!'));
    }

    res.json({
        status: 200,
        data: data
    });

});

}, function (err, req, res, next) {
    res.json({
        status: 501,
        message: err.message
    })
});

router.post('/login', function (req, res, next) {
    if (!req.body) {
        res.sendStatus(400);
    }

    let nickname = req.body.username;
    let password = req.body.password;

    const sql = "SELECT Id_User_med, Login, Password, Role FROM user_med WHERE Login
=?";

    let dataSql = [nickname];

    db.query(sql, dataSql, function (err, data, fields) {
        if (err) {
            next(new Error('Ошибка сервера!'));
        } else if (!data.length) {
            next(new Error('Введены некорректные данные!'));
        }
    });
}

```

```

    } else {
      bcrypt.compare(password, data[0].password).then((result) => {
        if (result === true) {
          const userData = {
            id: data[0].id,
            nickname: data[0].nickname,
            id_role: data[0].id_role,
          };
          const token = jwt.sign(userData, "23r34t33w4grthynjw5u6u45eye");
          userData.token = token;
          console.log(token);
          res.json({
            status: 200,
            message: 'access is allowed',
            user: userData
          })
        } else {
          next(new Error('Введены некорректные данные!'));
        }
      })
    }
  });
}, function (err, req, res, next) {
  res.json({
    status: 501,
    message: err.message
  })
});

module.exports = router;

index.js

var express = require('express');
```

```
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Express' });
});

module.exports = router;

users.js

var express = require('express');
var router = express.Router();

/* GET users listing. */
router.get('/', function(req, res, next) {
  res.send('respond with a resource');
});

module.exports = router;
```

Приложение Е

КОД СОЗДАНИЯ КЛИЕНТСКОЙ ЧАСТИ

(обязательное)

App.js

```
import logo from './logo.svg';
import React, { Component } from 'react';
import { Route, Switch, Redirect, withRouter } from "react-router-dom";
import './App.css';
import MainPage from "./pages/MainPage";
import MakeAppointment from "./pages/MakeAppointment";

class App extends Component {
  render() {
    return (
      <Switch>
        <Route exact path="/" component={MainPage}/>
        <Route exact path="/makeAppointment" component={MakeAppointment}/>
      </Switch>
    );
  }
}
```

export default withRouter(App);

API.js

```
import axios from "axios";
```

```
const instance = axios.create({
  baseURL: 'http://localhost:3001/api'
});
```

export default **instance**;

Store.js

```
import API from "./API"
```

```
class Store {
```

```

loginUser(loginData){
  API.post("/login", loginData).then(function (result){
    if (result.data["status"] === 200){
      console.log('succes');
    }
    else {
      console.log(result.data["message"]);
    }
  }).catch(function(){
    console.log('error');
  })
}

```

```

}

```

```

const store = new Store();
export default store;

Header.js

import React from 'react';
import PropTypes from 'prop-types';
import { makeStyles } from '@material-ui/core/styles';
import Toolbar from '@material-ui/core/Toolbar';
import Button from '@material-ui/core/Button';
import IconButton from '@material-ui/core/IconButton';
import SearchIcon from '@material-ui/icons/Search';
import Typography from '@material-ui/core/Typography';
import Link from '@material-ui/core/Link';
import store from "../store/Store";

```

```

const useStyles = makeStyles((theme) => ({

```

```

toolbar: {
  borderBottom: `1px solid ${theme.palette.divider}`,
},
toolbarTitle: {
  flex: 1,
},
toolbarSecondary: {
  justifyContent: 'space-between',
  overflowX: 'auto',
},
toolbarLink: {
  padding: theme.spacing(1),
  flexShrink: 0,
},
));

```

```

export default function Header(props) {
  const classes = useStyles();
  const { sections, title } = props;

  return (
    <React.Fragment>
      <Toolbar className={classes.toolbar}>
        <Typography
          component="h2"
          variant="h5"
          color="inherit"
          align="center"
          noWrap
          className={classes.toolbarTitle}
        >
          {title}

```

```

</Typography>
<IconButton>
  <SearchIcon />
</IconButton>
<Button variant="outlined" size="small" onClick={() => store.loginUser({})}>
  Войти
</Button>
</Toolbar>
<Toolbar component="nav" variant="dense" className={classes.toolbarSecondary}>
  {sections.map((section) => (
    <Link
      color="inherit"
      noWrap
      key={section.title}
      variant="body2"
      href={section.url}
      className={classes.toolbarLink}
    >
      {section.title}
    </Link>
  ))}
</Toolbar>
</React.Fragment>
);
}

```

```

Header.propTypes = {
  sections: PropTypes.array,
  title: PropTypes.string,
};

```

HeaderInfo.js

```
import React from 'react';
```

```

import PropTypes from 'prop-types';
import { makeStyles } from '@material-ui/core/styles';
import Paper from '@material-ui/core/Paper';
import Typography from '@material-ui/core/Typography';
import Grid from '@material-ui/core/Grid';
import Link from '@material-ui/core/Link';

const useStyles = makeStyles((theme) => ({
  mainFeaturedPost: {
    position: 'relative',
    backgroundColor: theme.palette.grey[800],
    color: theme.palette.common.white,
    marginBottom: theme.spacing(4),
    backgroundImage: 'url(https://source.unsplash.com/random)',
    backgroundSize: 'cover',
    backgroundRepeat: 'no-repeat',
    backgroundPosition: 'center',
  },
  overlay: {
    position: 'absolute',
    top: 0,
    bottom: 0,
    right: 0,
    left: 0,
    backgroundColor: 'rgba(0,0,0,.3)',
  },
  mainFeaturedPostContent: {
    position: 'relative',
    padding: theme.spacing(3),
    [theme.breakpoints.up('md')]: {
      padding: theme.spacing(6),
      paddingRight: 0,
    },
  },
}));

```

```

    },
  },
));

```

```

export default function HeaderInfo(props) {
  const classes = useStyles();
  const { post } = props;

  return (
    <Paper className={classes.mainFeaturedPost} style={{ backgroundImage:
    `url(${post.image})` }}>
      { /* Increase the priority of the hero background image */ }
      <img style={{ display: 'none' }} src={post.image} alt={post.imageText} />
      <div className={classes.overlay} />
      <Grid container>
        <Grid item md={6}>
          <div className={classes.mainFeaturedPostContent}>
            <Typography component="h1" variant="h3" color="inherit" gutterBottom>
              {post.title}
            </Typography>
            <Typography variant="h5" color="inherit" paragraph>
              {post.description}
            </Typography>
            <Link variant="subtitle1" href="#">
              {post.linkText}
            </Link>
          </div>
        </Grid>
      </Grid>
    </Paper>
  );
}

```

```
HeaderInfo.propTypes = {
  post: PropTypes.object,
};
```

MainPage.js

```
import React from 'react';
import {Component} from "react";

import {Button, Card, CardContent, Container, CssBaseline, Divider, Grid, Icon, Paper} from
"@material-ui/core";

import WebAssetIcon from '@material-ui/icons/WebAsset';
import Header from "../MainPageComponents/Header";
import HeaderInfo from "../MainPageComponents/HeaderInfo";
import Typography from "@material-ui/core/Typography";
import NotificationsActiveIcon from '@material-ui/icons/NotificationsActive';
```

```
export default class MainPage extends Component {
  render() {
    // куски логики

    const sections = [
      { title: 'Личный кабинет', url: '#' },
      { title: 'Медицинские организации', url: '#' },
      { title: 'Врачи', url: '#' },
      { title: 'Запись на медицинскую услугу', url: '/makeAppointment' },
      { title: 'Список медицинских услуг', url: '#' },
      { title: 'Онлайн-консультация', url: '#' },
      { title: 'Отзыв', url: '#' },
    ];

    const headerInf = {
      title: 'Услуга записи на прием к врачу',
```

description:

"Ваше здоровье в наших надежных руках!",

image: 'https://static.tildacdn.com/tild6238-3438-4339-b864-346363626463/pedde2.jpg',

imgText: 'main image description',

linkText: 'Новости портала медицинских услуг...!',

};

return(

<React.Fragment>

<CssBaseline />

<Container maxWidth="lg" style={{minHeight: '90vh'}}>

<Header title="Портал медицинских услуг города Пенза" sections={sections}/>

<main >

<HeaderInfo post = {headerInf} />

</main>

<div>

ДОБРО ПОЖАЛОВАТЬ!

Наш портал предназначен для записей на медицинские услуги.

Система позволяет записаться на прием к врачу или узнать расписание работы врачей, не выходя из дома или офиса.

Помимо записи вы можете просматривать все медицинские организации, всех врачей и их специальности, а также ознакомиться со списком медицинских услуг и их стоимостью.

Каждый пользователь сможет оставить отзыв после получения определенной услуги, а также сможет получить онлайн-консультацию врача не выходя из дома.

<Grid container spacing={3}>

<Grid item xs={6}>

<Card>

<CardContent>

<NotificationsActiveIcon/>

АКЦИИ КЛИНИК НА 17.06.2021 - 17.07.2021:

1) МЕДИКЛИНИК - Медицинская консультация для подготовки к лечению за рубежом (45 мин.) - 17.500 руб. (Вместо 20.500 руб)

2) ЗДОРОВЬЕ - Первичный прием врача хирурга - 500 РУБ. (Вместо 1000 Руб)

3) МЕДИЦИНСКИЙ ЦЕНТР ДИАГНОЗ - УЗИ желудка + толстого кишечника (обзорно) - 800 руб. (Вместо 1500 руб.)

4) СТОМАТОЛОГИЧЕСКАЯ КЛИНИКА "ЭНЕЯ" - Кабинетное (офисное) отбеливание на 2-х челюстях - 7.000 - руб. (Вместо 10.500 руб.)

</CardContent>

</Card>

</Grid>

<Grid item xs={6}>

<Card>

<CardContent>

<WebAssetIcon/>

ВАКАНСИИ КЛИНИК (Найдены во всех регионах)

На данный момент открытых вакансий нет...

</CardContent>

</Card>

</Grid>

</Grid>

</div>

</Container>

<Divider/>

<footer>

<Container maxWidth="lg">

<Typography variant="h6" align="center" gutterBottom>

г. Пенза

</Typography>

```

        <Typography variant="subtitle1" align="center" color="textSecondary"
component="p">

        2021

    </Typography>

</Container>

</footer>

</React.Fragment>

    )
}

```

```

}

```

MakeAppointment.js

```

import React from 'react';

import {Component} from "react";

import {Button, Card, CardContent, Container, CssBaseline, Divider, Grid, Icon, Paper} from
"@material-ui/core";

import CakeIcon from '@material-ui/icons/Cake';

import Header from "../MainPageComponents/Header";

import HeaderInfo from "../MainPageComponents/HeaderInfo";

import Typography from "@material-ui/core/Typography";

import HorizontalLinearStepper from "../MakeAppointmentNew/HorizontalLinearStepper";

export default class MakeAppointment extends Component {

  render() {

    // куски логики

    const sections = [

      { title: 'Личный кабинет', url: '#' },

      { title: 'Медицинские организации', url: '#' },

      { title: 'Врачи', url: '#' },

      { title: 'Запись на медицинскую услугу', url: '/makeApointment' },

      { title: 'Список медицинских услуг', url: '#' },

```

```

    { title: 'Онлайн-консультация', url: '#' },
    { title: 'Отзыв', url: '#' },
  ];

const headerInf = {
  title: 'Услуга записи на прием к врачу',
  description:
    "Ваше здоровье в наших надежных руках!",
  image: 'https://static.tildacdn.com/tild6238-3438-4339-b864-346363626463/pedde2.jpg',
  imgText: 'main image description',
  linkText: 'Данный сайт предназначен для...',
};

return(
  <React.Fragment>
    <CssBaseline />
    <Container maxWidth="lg" style={{ minHeight: '90vh' }}>
      <Header title="Портал медицинских услуг города Пенза" sections={sections}/>
      <main >
        <HeaderInfo post = {headerInf} />
      </main>
      <div>
        <HorizontalLinearStepper/>
      </div>
    </Container>
    <Divider/>
    <footer>
      <Container maxWidth="lg">
        <Typography variant="h6" align="center" gutterBottom>
          г. Пенза
        </Typography>

```

```

        <Typography variant="subtitle1" align="center" color="textSecondary"
component="p">
            2021
        </Typography>
    </Container>
</footer>
</React.Fragment>
)
}

```

```

}

```

HorizontalLinearStepper.js

```

import React, {useEffect} from 'react';
import { makeStyles } from '@material-ui/core/styles';
import Stepper from '@material-ui/core/Stepper';
import Step from '@material-ui/core/Step';
import StepLabel from '@material-ui/core/StepLabel';
import Button from '@material-ui/core/Button';
import Typography from '@material-ui/core/Typography';
import { MenuItem, TextField } from "@material-ui/core";
import API from "../store/API";

```

```

const useStyles = makeStyles((theme) => ({
  root: {
    width: '100%',
  },
  button: {
    marginRight: theme.spacing(1),
  },
  instructions: {
    marginTop: theme.spacing(1),
    marginBottom: theme.spacing(1),

```

```

    },
  }));

```

```

function getSteps() {
  return ['Личные данные', 'Данные приема', 'Подтверждение записи на медицинскую
услугу'];
}

```

```

export default function HorizontalLinearStepper() {
  const classes = useStyles();
  const [activeStep, setActiveStep] = React.useState(0);
  const [medicalFacility, setMedicalFacility] = React.useState([]);
  const [post, setPost] = React.useState([]);
  const [medicalWorker, setMedicalWorker] = React.useState([]);
  const steps = getSteps();

  useEffect(() => {
    getMedicalFacility();
    getPost();
    getMedicalWorker();
  }, []);
}

```

```

function getMedicalFacility() {
  API.post("/getMedicalFacility")
    .then(function (result) {
      setMedicalFacility(result.data["data"]);
      console.log(result.data["data"]);
    })
    .catch(function () {
      console.log("error");
    });
}

```

```
}
```

```
function getPost() {
  API.post("/getPost")
    .then(function (result) {
      setPost(result.data["data"]);
      console.log(result.data["data"]);
    })
    .catch(function () {
      console.log("error");
    });
}
```

```
function getMedicalWorker() {
  API.post("/getMedicalWorker")
    .then(function (result) {
      setMedicalWorker(result.data["data"]);
      console.log(result.data["data"]);
    })
    .catch(function () {
      console.log("error");
    });
}
```

```
function getStepContent(step) {
  switch (step) {
    case 0:
      return (
        <div>
          <form className={classes.root} noValidate autoComplete="off">
            <div>
```

```

    <TextField style={{width: 200}} id="standard-basic1" label="Фамилия" />
    &nbsp;

    <TextField id="standard-basic2" label="Имя" /> &nbsp;

    <TextField id="standard-basic3" label="Отчество" />

  </div>

  <div>

    <TextField id="standard-basic4" label="Телефон" />

  </div>

  <div>

    <TextField id="standard-basic5" label="Почта" />

  </div>

</form></div>;

```

case 1:

```

return (<div> <form className={classes.root} noValidate autoComplete="off">
  <div>
    <TextField
      id="standard-select-currency1"
      select
      label="Медицинское учреждение"
      //value={currency}
      //onChange={handleChange}
      helperText="Пожалуйста, выберите медицинское учреждение"
    >
      {medicalFacility.map((option) => (
        <MenuItem key={option.Id_Medical_facility}
value={option.Id_Medical_facility}>
          {option.Name_medical_facility}
        </MenuItem>
      ))}
    </TextField>
  </div>
  <div>
    <TextField

```

```

        id="standard-select-currency2"

        select

        label="Специальность"

        //value={currency}

        //onChange={handleChange}

        helperText="Пожалуйста, выберите специальность"
    >

    {post.map((option) => (
        <MenuItem key={option.Id_Post} value={option.Id_Post}>
            {option.Name_post}
        </MenuItem>
    ))}
</TextField>
</div>
<div>
    <TextField
        id="standard-select-currency3"

        select

        label="Врач"

        //value={currency}

        //onChange={handleChange}

        helperText="Пожалуйста, выберите врача"
    >

    {medicalWorker.map((option) => (
        <MenuItem key={option.Id_Medical_worker}
value={option.Id_Medical_worker}>
            {option.Surname}
        </MenuItem>
    ))}
</TextField>
</div>
<div>

```

```

    <form className={classes.container} noValidate>
      <TextField
        id="datetime-local"
        label="Next appointment"
        type="datetime-local"
        defaultValue="2017-05-24T10:30"
        className={classes.textField}
        InputLabelProps={{
          shrink: true,
        }}
      />
    </form>
  </div>
</form>
</div>);

```

case 2:

```

    return 'Запись на прием успешно подтверждена! Ожидаем вас в больнице №6, в
    кабинете №1 у терапевта Бочкарева 28.06.2021 в 10:30';

```

default:

```

    return 'Unknown step';
  }
}

```

```

const handleNext = () => {
  setActiveStep((prevActiveStep) => prevActiveStep + 1);
};

```

```

const handleBack = () => {
  setActiveStep((prevActiveStep) => prevActiveStep - 1);
};

```

```

const handleReset = () => {

```

```

    setActiveStep(0);
  };

  return (
    <div className={classes.root}>
      <Stepper activeStep={activeStep}>
        {steps.map((label, index) => {
          const stepProps = {};
          const labelProps = {};

          return (
            <Step key={label} {...stepProps}>
              <StepLabel {...labelProps}>{label}</StepLabel>
            </Step>
          );
        })}
      </Stepper>
      <div>
        {activeStep === steps.length ? (
          <div>
            <Typography className={classes.instructions}>
              All steps completed - you're finished
            </Typography>
            <Button onClick={handleReset} className={classes.button}>
              Reset
            </Button>
          </div>
        ) : (
          <div>
            {getStepContent(activeStep)} <br/>
          </div>

```

```

        <Button disabled={activeStep === 0} onClick={handleBack}
className={classes.button}>
            Назад
        </Button>

        <Button
            variant="contained"
            color="primary"
            onClick={handleNext}
            className={classes.button}
        >
            {activeStep === steps.length - 1 ? 'Завершить' : 'Далее'}
        </Button>
    </div>
</div>
    )}
</div>
</div>
);
}

```