Министерство науки и высшего образования РФ

Федеральное государственное автономное

образовательное учреждение

высшего образования

«**СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**»

Космических и информационных технологий

институт

Систем искусственного интеллекта

кафедра

**КУРСОВАЯ РАБОТА**

АИС “Запись на прием к врачу”

тема работы

Преподаватель \_\_\_\_\_\_\_\_\_\_ Д.А. Коченов

подпись, дата инициалы, фамилия

Студент КИ22-21б 032211762 \_\_\_\_\_\_\_\_\_\_ И.Д. Курочкин

номер группы номер зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2024

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc170332908)

[1 Проектирование авторизованной информационной системы 4](#_Toc170332909)

[1.1 Анализ существующего программного обеспечения 4](#_Toc170332910)

[1.2 Разработка концептуальной модели АИС 5](#_Toc170332911)

[1.3 Разработка логической модели ИС 8](#_Toc170332912)

[1.4 Выбор целевой СУБД 23](#_Toc170332913)

[1.5 Разработка логической модели предметной области 24](#_Toc170332914)

[2 Разработка программного продукта 32](#_Toc170332915)

[2.1 Структура программного продукта 32](#_Toc170332916)

[2.2 Реализация бизнес правил 34](#_Toc170332917)

[2.3 Руководство пользователя 36](#_Toc170332918)

[2.4 Руководство работника 43](#_Toc170332919)

[2.5 Тестирование программного продукта 44](#_Toc170332920)

[ЗАКЛЮЧЕНИЕ 46](#_Toc170332921)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 47](#_Toc170332922)

[ПРИЛОЖЕНИЕ А 48](#_Toc170332923)

# ВВЕДЕНИЕ

В современном медицинском обслуживании эффективная организация записи на прием к врачу играет ключевую роль в обеспечении качественного и своевременного обслуживания пациентов. Автоматизированные информационные системы (АИС) для записи на прием становятся неотъемлемой частью современных здравоохранительных учреждений, обеспечивая удобство и доступность для пациентов и медицинского персонала. Разработка и оптимизация таких систем требуют глубокого понимания требований пользователя, специфики медицинских процессов и технических возможностей. Анализ существующих решений в этой области позволяет выявить лучшие практики и применить их для создания инновационных и адаптированных под специфику медицинской среды информационных систем.

Предмет курсовой работы: технологии и методы разработки АИС “Запись на прием к врачу”, включая архитектуру, функциональные возможности, алгоритмы работы и аспекты безопасности.

Объект курсовой работы: информационная система, предназначенная для записи на прием к врачу, а также управления записями на прием к врачу в режиме реального времени.

Цели:

1. Анализ предметной области – основных требований, технологий и средств разработки АИС для записи на прием к врачу.
2. Изучение основ создания АИС для записи на прием к врачу в режиме реального времени.
3. Программная реализация АИС с учетом поставленных требований.

Задачи:

1. Описание предметной области разрабатываемой системы, включая особенности процесса записи на прием к врачу и требования пользователей.
2. Определение требований и критериев для оценки надежности и качества разрабатываемой системы.
3. Выбор технологий и средств разработки, наиболее подходящих для реализации целей и задач проекта.
4. Проектирование информационной системы, включая описание жизненного цикла пользователя и основные этапы работы приложения.
5. Реализация системы с использованием выбранных технологий, включая разработку функциональности для эффективной записи на прием к врачу.
6. Тестирование системы на различных уровнях, включая модульное тестирование отдельных компонентов, интеграционное тестирование системы в целом и проверку на соответствие требованиям пользователя.

# 1 Проектирование авторизованной информационной системы

## 1.1 Анализ существующего программного обеспечения

Для анализа аналогов автоматизированной информационной системы (АИС) "Запись на прием к врачу" рассмотрим несколько существующих решений, которые обеспечивают подобные функции. Основное внимание уделим их функционалу, преимуществам и недостаткам.

а) Поликлиника.ру:

1. Функционал:

* Запись на прием к врачу через интернет и мобильное приложение.
* Просмотр расписания врачей.
* Возможность отмены и переноса записи.
* Напоминания о приеме по СМС и электронной почте.
* Личный кабинет пациента с историей посещений и медицинскими данными.

1. Преимущества:

* Удобный интерфейс и простота использования.
* Возможность записи в любое время суток.
* Напоминания снижают вероятность пропуска приема.

1. Недостатки:

* Возможны технические сбои в работе системы.
* Ограниченная интеграция с медицинскими информационными системами (МИС) некоторых учреждений.
* Не все медицинские учреждения поддерживают онлайн-запись.

б) Доктор на работе:

1. Функционал:

* Онлайн-запись на прием к врачу.
* Подбор врачей по специализации и отзывам пациентов.
* Напоминания о приеме.
* Возможность оставить отзыв о враче.

1. Преимущества:

* Широкий выбор врачей и клиник.
* Подробная информация о врачах и их рейтингах.
* Удобный поиск по специализациям и локации.

1. Недостатки:

* Отсутствие интеграции с электронными медицинскими картами.
* Возможность записи ограничена партнёрскими клиниками.
* Интерфейс может быть перегружен информацией.

в) ЕМИАС (Единая медицинская информационно-аналитическая система):

1. Функционал:

* Единая система записи на прием к врачу для всех государственных поликлиник.
* Просмотр расписания врачей.
* Интеграция с электронными медицинскими картами.
* Возможность записи через интернет, мобильное приложение.

1. Преимущества:

* Центральная система для всех государственных учреждений.
* Интеграция с другими медицинскими системами.
* Возможность записи через различные каналы.

1. Недостатки:

* Высокая нагрузка на систему может приводить к сбоям.
* Ограниченная функциональность по сравнению с частными сервисами.
* Требует регистрации в системе и подтверждения через госуслуги.

г) Яндекс.Здоровье:

1. Функционал:

* Онлайн-запись на прием к врачам партнерских клиник.
* Консультации с врачами онлайн.
* Электронные рецепты и анализы.
* Личный кабинет с историей обращений.

1. Преимущества:

* Возможность онлайн-консультаций.
* Удобная интеграция с другими сервисами Яндекса.
* Современный и интуитивно понятный интерфейс.

1. Недостатки:

* Ограничено партнёрскими клиниками.
* Некоторые услуги платные.
* Не все функции доступны в регионах.

Эти системы представляют различные подходы к организации записи на прием к врачу, предлагая различные уровни интеграции, удобства и функциональности. Основываясь на анализе этих аналогов, можно выделить ключевые аспекты, которые необходимо учесть при разработке собственной АИС для записи на прием к врачу: удобство интерфейса, интеграция с медицинскими информационными системами, напоминания о приеме, возможность онлайн-консультаций и гибкость в выборе каналов записи.

## 1.2 Разработка концептуальной модели АИС

* + 1. Краткое описание сути выполняемого концептуального проектирования

Концептуальное проектирование автоматизированной информационной системы (АИС) заключается в формировании абстрактного представления данных, которое отражает основные элементы и их взаимосвязи в предметной области. Основная цель концептуального проектирования — разработка модели данных, определение ключевых сущностей, их атрибутов и связей между ними.

* + 1. Создание ER-модели

ER-модель (модель "сущность-связь") представляет собой графическое изображение сущностей (таблица 1.1) и их связей (таблица 1.2). ER-модель представлена на рисунке 1.2.1.

Таблица 1.1 – Сведения о типах сущностей АИС

|  |  |
| --- | --- |
| **Имя сущности** | **Описание** |
| Пользователь | Личность, использующая систему для записи на прием к доктору и оставляющая отзывы. |
| Доктор | Специалист, оказывающий медицинские услуги и принимающий записи от пользователей. |
| Запись к врачу | Запись на прием к доктору, сделанная пользователем. |
| Отзывы | Отзывы пользователей о докторах и услугах. |
| Заявка на звонок | Заявка на звонок, сделанная пользователем для связи с доктором. |
| Верификационные данные | Данные для верификации пользователя. |
| Новости | Информационная статья, написанная доктором. |

Таблица 1.2 – Сведения о типах связи АИС

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя связи** | **Степень связи** | **Сущность 1** | **Сущность 2** | **Описание связи** |
| Записывается на прием | 1 × 𝑛 | Пользователь | Запись к врачу | Один пользователь может записаться на много приемов, каждая запись относится только к одному пользователю. |
| Принимает запись | 1 × 𝑛 | Доктор | Запись к врачу | Один доктор может принимать много записей, каждая запись связана только с одним доктором. |
| Может иметь | 1 × 𝑛 | Пользователь | Отзывы | Один пользователь может оставить много отзывов, каждый отзыв связан только с одним пользователем. |
| Может получать | 1 × 𝑛 | Доктор | Отзывы | Один доктор может получать много отзывов, каждый отзыв относится только к одному доктору. |
| Может делать | 1 × 𝑛 | Пользователь | Заявка на звонок | Один пользователь может сделать много заявок на звонок, каждая заявка связана только с одним пользователем. |
| Может иметь | 1 × 1 | Пользователь | Верификационные данные | Один пользователь может иметь только один набор верификационных данных. |
| Может просматривать | m × 𝑛 | Пользователь | Новость | Много пользователей могут просматривать много новостей. |

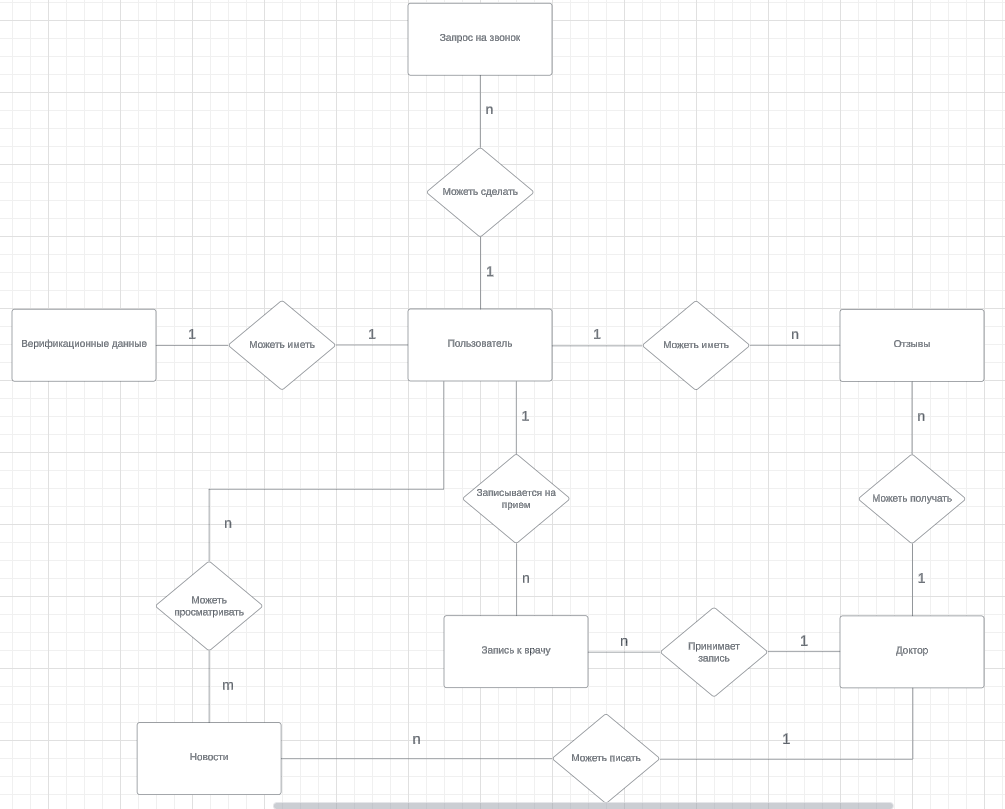


Рисунок 1.2.1 – ER-модель АИС

## 1.3 Разработка логической модели ИС

* + 1. Разработка диаграммы *Use Case*

Диаграмма *Use Case* представляет собой графическое отображение взаимодействий пользователей с системой. Она служит для визуализации функциональных требований и описания основных сценариев использования приложения. Диаграмма представлена на рисунке 1.3.1.

Таблица 1.2 – Варианты использования АИС

|  |  |  |  |
| --- | --- | --- | --- |
| *Код* | *Основной актор* | *Наименование* | *Формулировка* |
| P1 | Пациент, Администратор | Создание записи | Пациент может создать запись путем заполнения формы на сайте. Администратор может создать запись путем прямого ее добавления в базу данных. |
| P2 | Пациент | Оставление обратной связи пациентом | Пациент может оставить обратную связь о посещении через форму, присланную ему чат-ботом посредством мессенджера WhatsApp после посещения. |
| P3 | Пациент, Администратор | Просмотр информации о записи | Пациент с помощью чат-бота может получить актуальную информацию о грядущей записи, а также о всех записях, которые он посетил. Администратор может просмотреть все записи к врачам |
| P4 | Пациент, Администратор | Отмена записи | Пациент может отменить одну или несколько грядущих записей с помощью чат-бота. При необходимости администратор может удалить запись, тем самым отменить запись к врачу. |
| P5 | Пациент | Выбор даты и времени посещения врача | Во время создания записи, пациент должен выбрать свободное время для посещения врача. |
| P6 | Пациент | Выбор специализации врача | Во время создания записи, пациент должен выбрать специализацию врача, к которому он хочет записаться. |
| P7 | Пациент | Выбор врача | Во время создания записи, пациент может выбрать конкретного врача, к которому он хочет попасть. |
| D1 | Врач | Сообщение о присутствии на приеме | Врач должен отследить и подтвердить или опровергнуть фактическое присутствие пациента на приеме путем проставления/не проставления галочки в БД |
| D2 | Врач | Просмотр записей | Врач может просмотреть все записи на свое имя на текущий день, неделю и месяц. |
| С1 | Главный врач | Создание отчета | Главный врач может с помощью несложных манипуляций создать итоговый отчет по загруженности и посещаемости каждого врача. |
| С2 | Главный врач | Обратная связь | Главный врач может связаться с пациентом, оставившим плохой отзыв и более детально уточнить обстоятельства произошедшего |

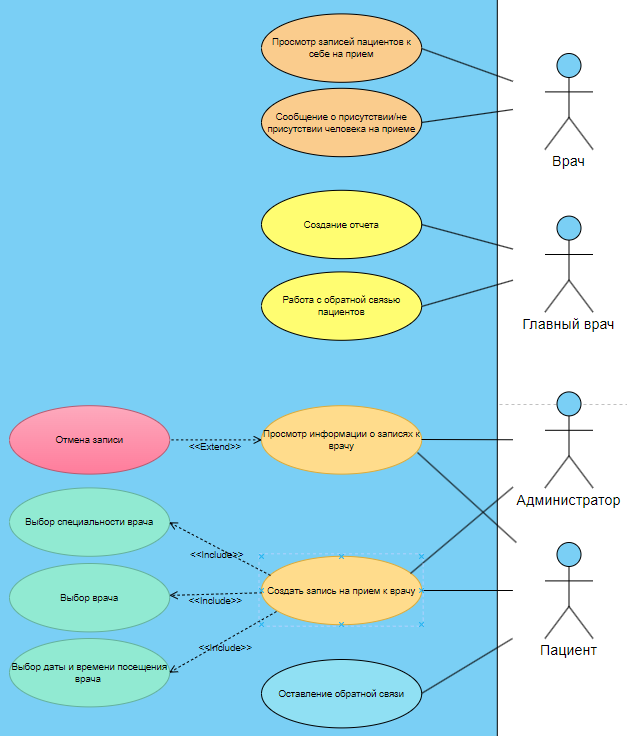


Рисунок 1.3.1 – Диаграмма Use Case

* + 1. Разработка диаграммы деятельности (*Activity Diagram*)

Диаграмма активности представляет собой графическое отображение последовательности действий или потоков управления в системе. Она используется для моделирования различных процессов и их логических потоков, помогая понять, как система обрабатывает операции и реагирует на события.

Наиболее существенным прецедентом, который нуждается в полном описании процесса, является запись на прием к врачу.

Базовый поток - Создание записи на прием к врачу без направления

1. Пациент выбирает специализацию врача.
2. На основании специализации врача система выводит новый выпадающий список, содержащий ФИО врачей, имеющих специализацию, выбранную в прошлом пункте.
3. Пациент выбирает ФИО врача.
4. На основании ФИО врача система выводит календарь, в котором отображен текущий месяц. По желанию пользователь может переключиться на следующие месяцы. Пациент может выбрать только дату, где имеется как минимум одна свободная запись к врачу, остальные записи недоступны к выбору.
5. Пациент выбирает дату посещения врача.
6. На основании ФИО врача и даты посещения система выводит все свободные окна в расписании врача.
7. Пациент выбирает время посещения врача.
8. Система разблокирует кнопку “Записаться к врачу”.
9. Пациент нажимает на кнопку “Записаться к врачу”.

Альтернативный поток *-* Наличие направление к специалисту.

1. Пациент нажимает на кнопку “У меня есть направление к врачу”.
2. Система проверяет есть ли у человека направление к врачу по его паспортным данным.
3. После того, как система удостоверяется в наличии направления, алгоритм переходит ко второму этапу базового потока.
4. Если направления не было найдено, то система выводит ошибку и переходит к первому этапу базового потока.

Диаграмма деятельности представлена на рисунке 1.3.2.

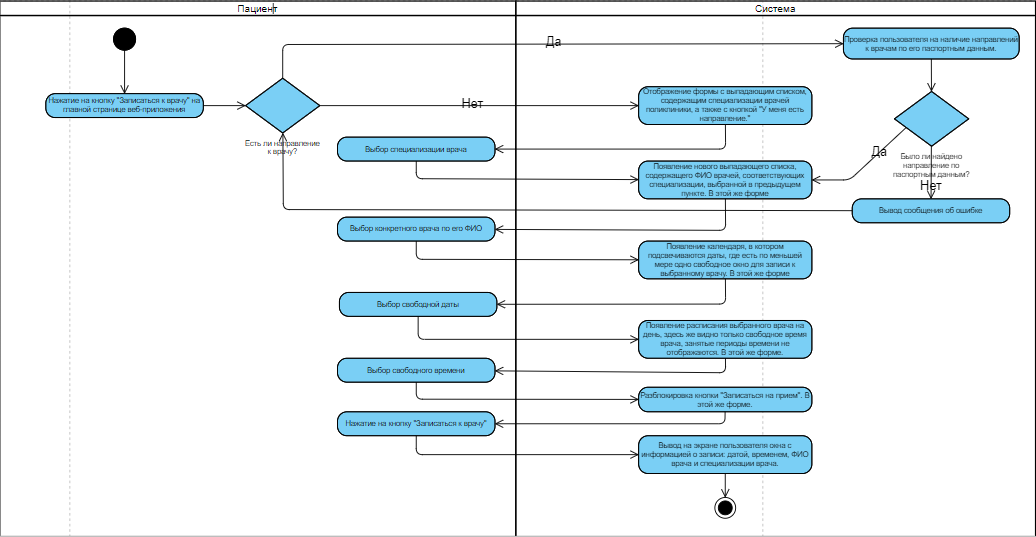


Рисунок 1.3.2 – Диаграмма деятельности

1.3.3 Разработка диаграммы классов

Предметная область автоматизированной информационной системы представлена пятью сущностями, выраженными классами. Классы Doctor, Patient и Admin являются потомками класса User, соответственно между ними связь наследования. Также оба эти класса находятся в отношении ассоциации с классом Appointment, так как используют его только в параметрах методов. Класс Schedule находится в агрегации с классом Appointment, причем Appointment может содержать лишь один экземпляр Schedule (для каждой записи существует только одно расписание), а Appointment может содержаться во многих экземпляров Schedule. Диаграмма классов изображена на рисунке 1.3.3.

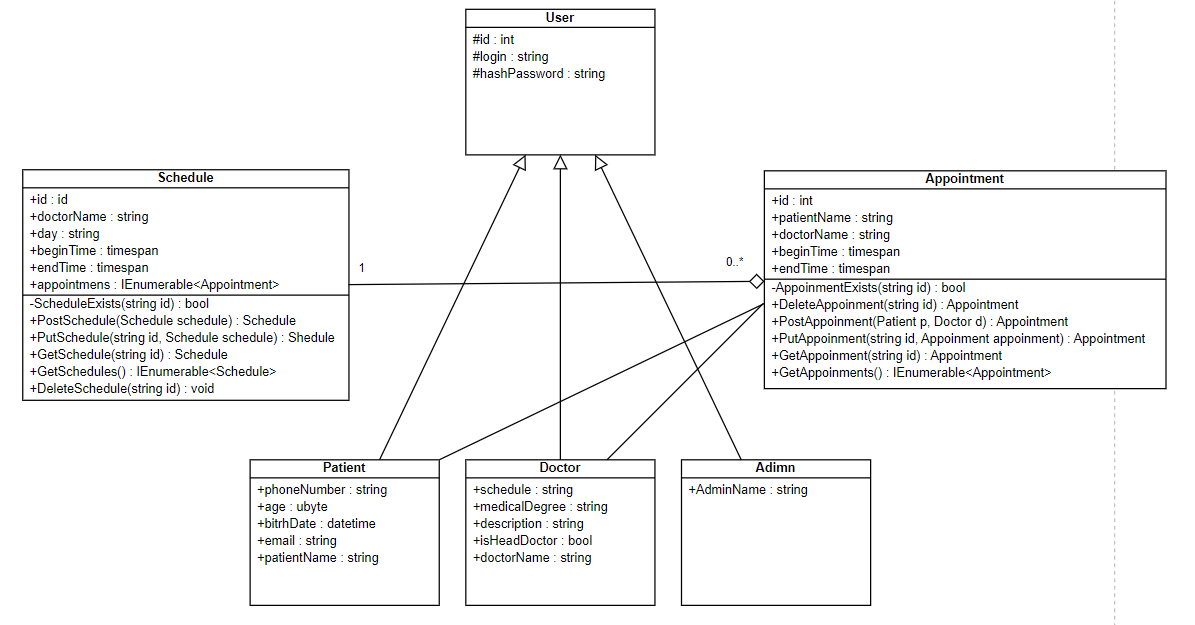


Рисунок 1.3.3 – Диаграмма классов

1.3.4 Разработка диаграмм последовательности и кооперации

*1.3.4.1 Описание диаграмм последовательности*

Диаграмма последовательности используется для моделирования временной последовательности взаимодействий, показывая, как процессы развиваются с течением времени.

Рассмотрим прецедент, изображенный на рисунке 1.3.4.1:

1. Когда клиент нажимает на кнопку “Записаться к врачу” (предполагается, что клиент уже авторизован в системе), вызывается действие 1.1. Это действие является обработчиком события для представления. После действие вызывает метод 1.2 в классе Doctor, который обращается к БД и возвращает массив специализаций врачей. Действие 1.3 передает эти данные от сервера в JSON формате на клиент (View). Действие 1.4 отображает полученные данные в формате выпадающего списка для клиента.
2. Клиент выбирает специализацию врача в выпадающем списке, после чего срабатывает действие 1.5, которое обрабатывает событие для представления и извлекает из него выбранную пользователем специализацию. После вызывается метод 1.6 в классе Doctor, который отправляет запрос к БД, передавая в параметрах строки выбранную клиентом специализацию. Действие 1.7 возвращает View массив объектов класса Doctor, которые имеют специализацию, указанную клиентом. Действием 1.8 View отображает полученные данные в формате выпадающего списка для клиента.
3. Клиент выбирает имя подходящего ему доктора в выпадающем списке, после чего срабатывает действие 1.9, которое обрабатывает событие для представления и извлекает из него имя выбранного пользователем доктора. После вызывается метод 1.10 в классе Doctor, который получает все объекты типа Schedule (Schedule – расписание на один конкретный рабочий день, тогда []Schedule – все расписание доктора). Действие 1.11 возвращает полученные объекты Schedule во View. Действием 1.12 View генерирует с помощью метода ShowSchedule пользовательский вывод для клиента и передает его на клиент.
4. Клиент выбирает дату и время записи (выбрать можно только свободное время, т.е. промежуток времени, где isAvailable = true) и нажимает на кнопку “Записаться к врачу”, вызывается действие 1.13. Это действие является обработчиком события для представления, оно берет все данные из формы, и передает их в метод 1.14 PostAppointment. PostAppointment создает новую запись к врачу и возвращает done. (поскольку appointment является методом класса Patient, передавать объект Patient не требуется)

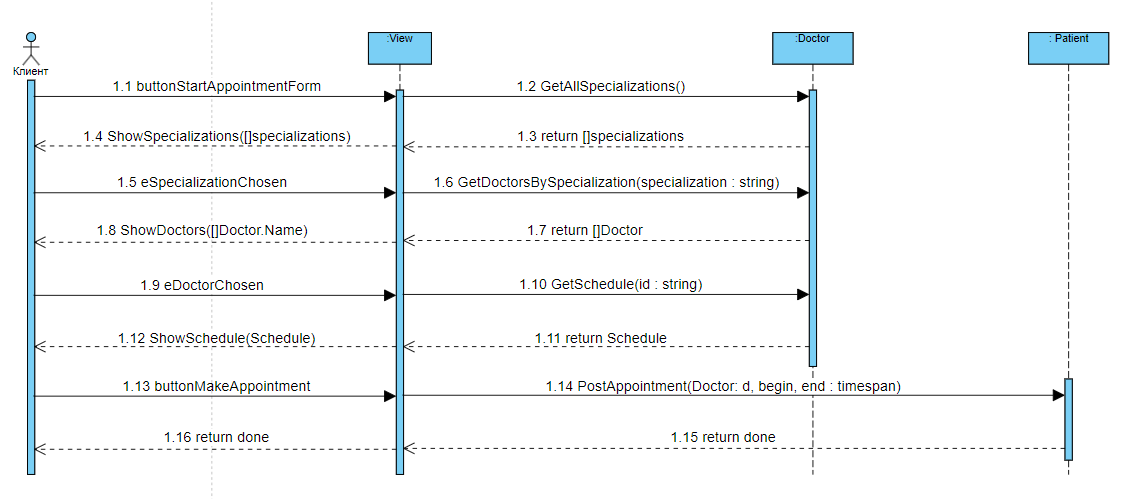


Рисунок 1.3.4.1 – Диаграмма Последовательности

*1.3.4.2 Описание диаграммы кооперации*

Диаграмма кооперации фокусируется на структурных аспектах взаимодействий, подчеркивая взаимосвязи и сотрудничество между участниками процесса.

В связи с тем, как устроена архитектура приложения, диаграмма кооперации очень схожа с диаграммой последовательности. Однако по сравнению с последней она позволяет анализировать и определять взаимодействия и зависимости между объектами и классами в системе в целом, то, как они кооперируются, упуская из виду, в какой последовательности происходит взаимодействие. В свою очередь, это помогает обеспечить общее понимание архитектуры системы.

Рассмотрим диаграмму кооперации для части диаграммы UseCase, связанной с актором Клиент, а именно следующие варианты использования: создание записи на прием к врачу, просмотр информации о записях к врачу и отмена записи. Диаграммы кооперации на рисунках 1.3.4.1 и 1.3.4.2.

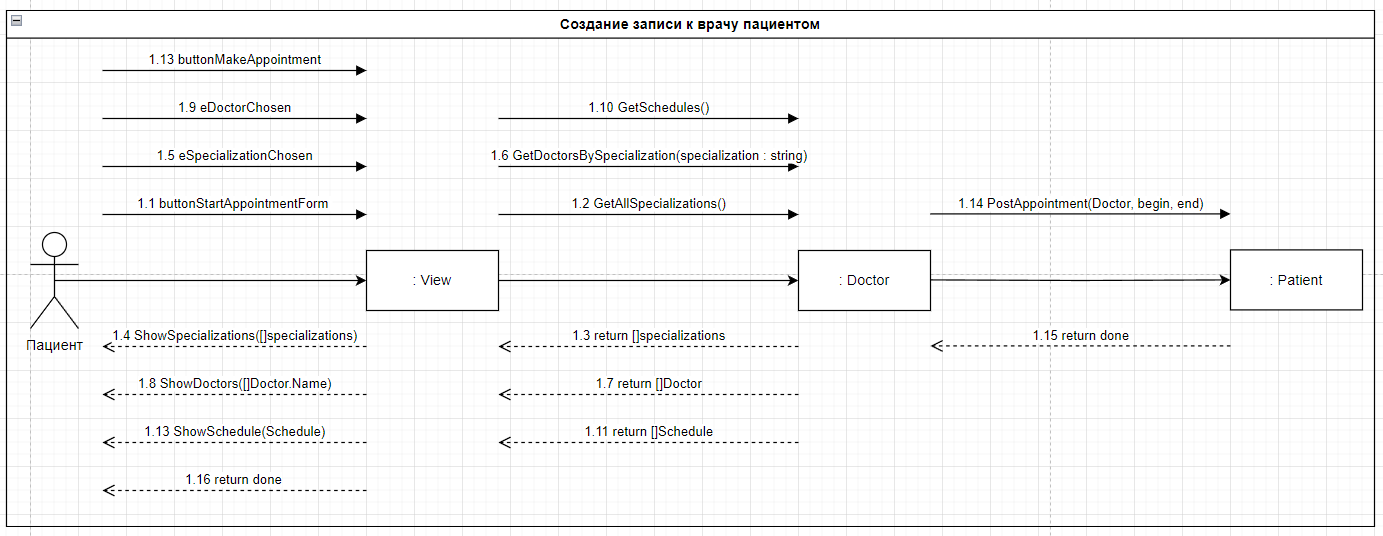


Рисунок 1.3.4.2.1 – Диаграмма кооперации 1

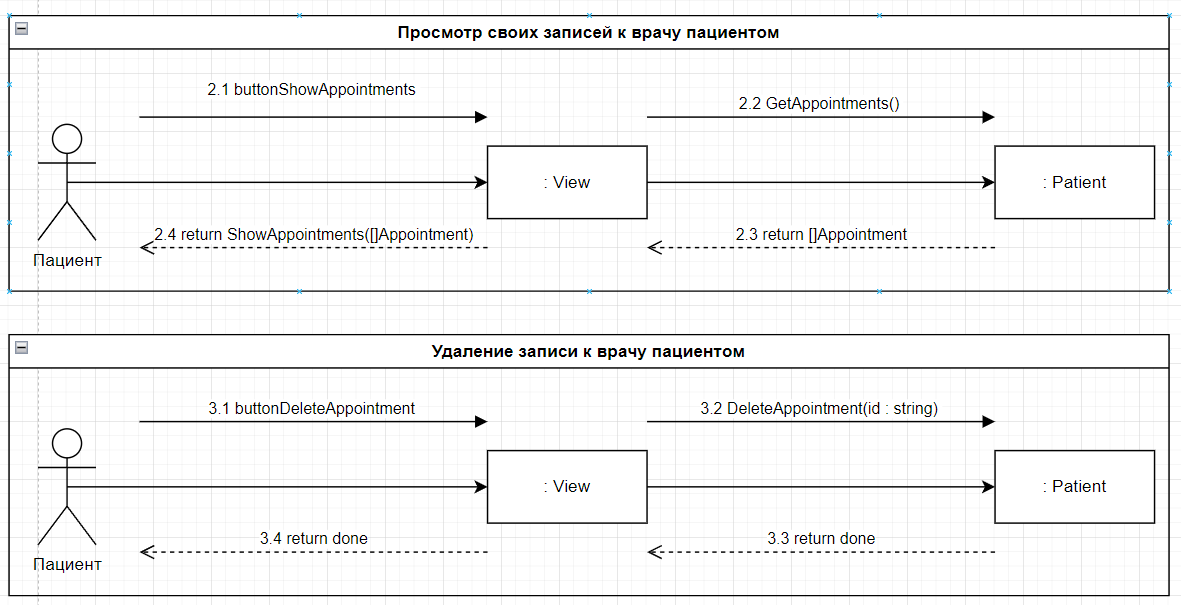


Рисунок 1.3.4.2.2 – Диаграмма кооперации 2

1.3.5 Разработка диаграммы состояний

Диаграмма состояний представляет собой графическое отображение различных состояний объекта в системе и переходов между этими состояниями. Она используется для моделирования жизненного цикла объектов, показывая, как они изменяются в ответ на внутренние и внешние события.

Экземпляр класса Appointment имеет 5 состояний (рисунок 1.3.5):

1. Свободна: возникает при создании свободной записи в базе данных;
2. Занята: возникает при событии, когда пациент записывается к врачу. Поле isAvailable принимает значение false;
3. Состоялась: возникает, когда время записи прошло и пациент посетил врача. Поле isCompleted принимает значение true;
4. Не состоялась (не пришел пациент): возникает в случае, если пациент не отменил запись и не пришел в назначенное время. Поле isCompleted принимает значение false;
5. Не состоялась (запись не была занята, время записи прошло): возникает, если запись не была занята и время записи прошло. Поле isExpired принимает значение true;



Рисунок 1.3.5 – Диаграмма состояний экземпляра класса Appointment

1.3.6 Разработка диаграмм компонентов и развертывания

*1.3.6.1 Разработка диаграммы компонентов*

На диаграмме (рисунки 1.3.6.1.1 и 1.3.6.1.2) представлено два основных компонента: Client и Server, которые описывают логику и организацию работы представления и серверной части соответственно.

1. Client

* GUI: Графический пользовательский интерфейс. Взаимодействует с компонентом Client Scripts.
* Client Scripts: Скрипты, работающие на стороне клиента (например, JavaScript коды). Они взаимодействуют с сервером приложений через компонент App Server и сервер авторизации через компонент Auth Server.

2. Server

* App Server: Основной сервер приложений, обрабатывающий маршруты и бизнес-логику.
  + Routes: Компонент, отвечающий за маршрутизацию запросов. Направляет запросы к соответствующим контроллерам.
  + DoctorController: Контроллер, обрабатывающий запросы, связанные с докторами.
    - Doctor: Модуль, управляющий данными докторов.
    - Appointment: Модуль, управляющий данными записей на прием к врачам.
  + UserController: Контроллер, обрабатывающий запросы, связанные с пользователями.
    - Users: Модуль, управляющий данными пользователей.
    - Review: Модуль, управляющий отзывами пользователей.
    - Appointment: Модуль, управляющий данными записей на прием пользователей.
    - News: Модуль, управляющий новостями.
* Auth Server: Сервер авторизации, который обрабатывает запросы на аутентификацию и авторизацию пользователей.

3. DataBase Server

* DataBase Server: Сервер базы данных, где хранятся данные.
  + IDoctorDB: База данных докторов.
  + IUserDB: База данных пользователей.
  + IAuthDB: База данных авторизации.

4. Services

* DoctorServices: Сервис, предоставляющий бизнес-логику для работы с данными докторов. Взаимодействует с IDoctorDB.
* UserServices: Сервис, предоставляющий бизнес-логику для работы с данными пользователей. Взаимодействует с IUserDB.

Взаимодействия компонентов:

1. Client взаимодействует с App Server через маршруты, которые направляют запросы к соответствующим контроллерам (DoctorController и UserController).
2. App Server обрабатывает запросы через свои контроллеры (DoctorController и UserController), которые затем взаимодействуют с соответствующими сервисами (DoctorServices и UserServices).
3. DoctorServices и UserServices обращаются к базам данных (IDoctorDB и IUserDB) для выполнения операций с данными.
4. Client также взаимодействует с Auth Server для выполнения операций аутентификации и авторизации.
5. Auth Server взаимодействует с IAuthDB для проверки учетных данных и токенов пользователей.

Общее описание:

Диаграмма показывает четкую разделенность между клиентской и серверной частью приложения, а также между логикой приложения и данными.

Это обеспечивает модульность и упрощает управление и развитие системы.

* Клиентская часть включает в себя интерфейс и клиентские скрипты, которые обрабатывают взаимодействие пользователя и передают запросы на сервер.
* Серверная часть состоит из сервера приложений и сервера авторизации, которые обрабатывают соответствующие запросы и взаимодействуют с базами данных.
* Базы данных хранят информацию о пользователях, докторах и записях на прием, а также данные для авторизации.

Такая архитектура позволяет легко добавлять новые функции и компоненты, обеспечивая при этом безопасность и производительность системы.

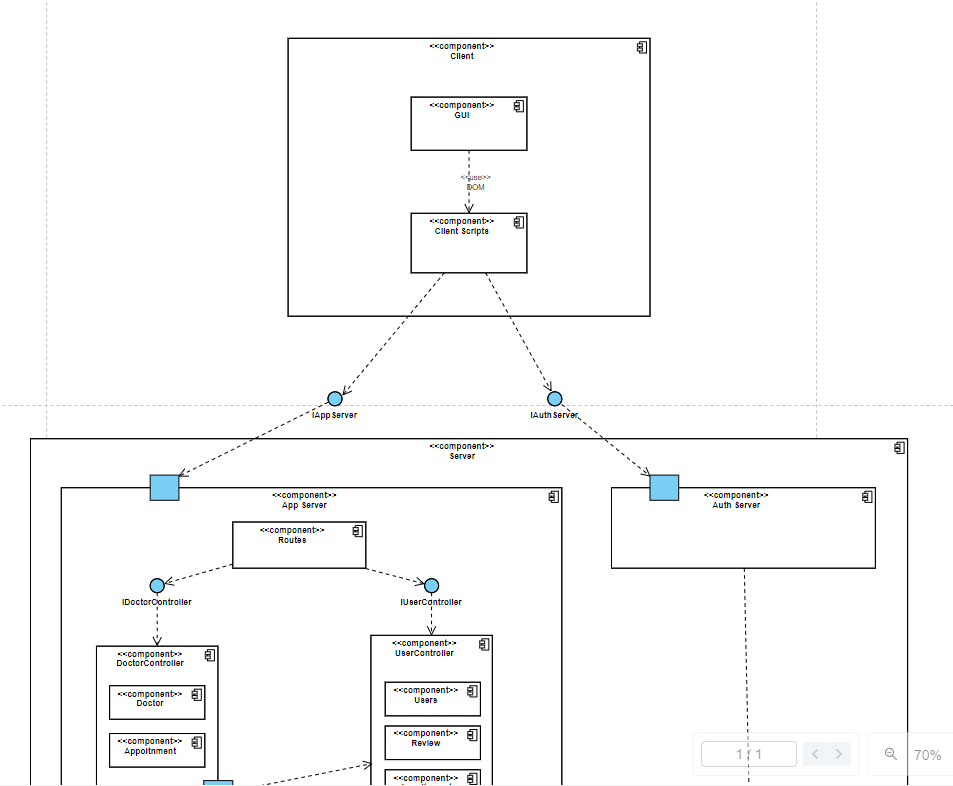


Рисунок 1.3.6.1.1 – Диаграмма компонентов 1

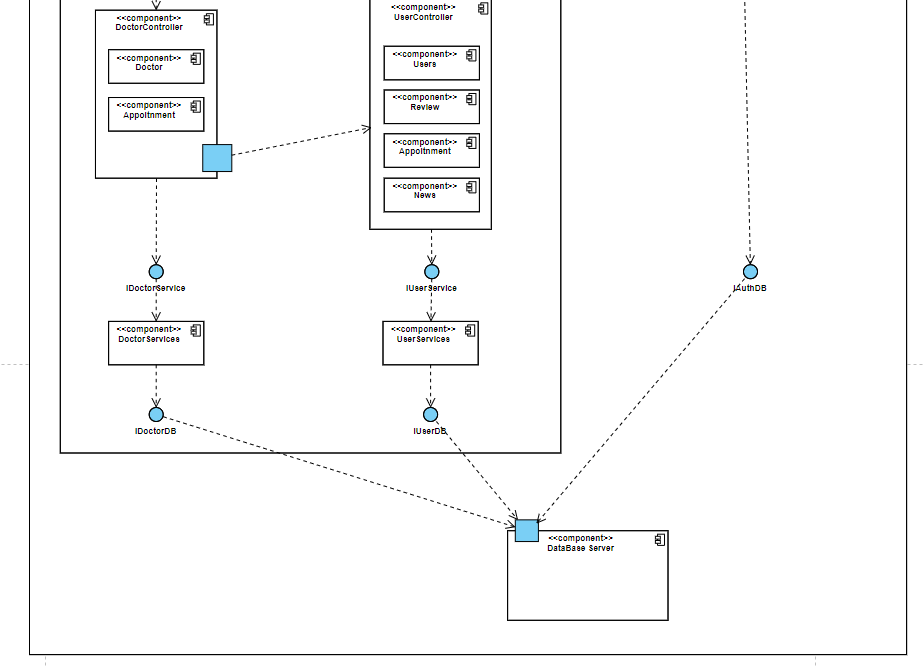


Рисунок 1.3.6.1.2 – Диаграмма компонентов 2

*1.3.6.2 Разработка диаграммы развертывания*

Диаграмма развертывания помогает понять физическую архитектуру АИС “запись на прием к врачу”, а также взаимодействия между различными компонентами и узлами. Диаграмма отображена на рисунке 1.3.6.2.

1. Client Machine (Клиентская машина)
   * Описание: Это устройство, используемое пользователями для доступа к системе. Оно включает в себя веб-браузер, скрипты клиента и графический пользовательский интерфейс (GUI).
   * Компоненты:
     + Browser (Браузер): Приложение, используемое для доступа к веб-приложению. Оно отправляет HTTP-запросы к веб-серверу и отображает ответы.
     + Client Scripts (Клиентские скрипты): JavaScript или другие скрипты, выполняемые на стороне клиента. Они позволяют динамически обновлять контент на странице без перезагрузки.
     + GUI (Графический пользовательский интерфейс): Интерфейс, с которым взаимодействует пользователь. Он представляет собой веб-страницы, созданные с использованием HTML, CSS и JavaScript.
2. Web Server (Веб-сервер)
   * Описание: Сервер, обрабатывающий HTTP-запросы от клиентов и взаимодействующий с базами данных и сервером авторизации. Основной компонент здесь - сервер приложений.
   * Компоненты:
     + App Server (Сервер приложений): Обрабатывает входящие HTTP-запросы, выполняет бизнес-логику и генерирует ответы. Включает маршруты для направления запросов к соответствующим контроллерам и бизнес-логике.
3. Auth Server (Сервер авторизации)
   * Описание: Сервер, который управляет аутентификацией и авторизацией пользователей. Он обрабатывает запросы на вход в систему и проверку токенов.
   * Компоненты:
     + Auth Server (Сервер авторизации): Обрабатывает запросы на аутентификацию, генерирует и проверяет токены доступа, а также взаимодействует с базой данных авторизации для хранения и проверки учетных данных.
4. Database Server (Сервер базы данных)
   * Описание: Сервер, на котором хранятся все данные системы, включая информацию о докторах, пользователях и аутентификации. Он взаимодействует с веб-сервером и сервером авторизации для предоставления необходимых данных.
   * Компоненты:
     + DoctorDB (База данных докторов): Хранит информацию о докторах, включая их профили, расписания и записи на прием.
     + UserDB (База данных пользователей): Хранит информацию о пользователях системы, включая их профили и отзывы.
     + AuthDB (База данных авторизации): Хранит учетные данные пользователей, токены аутентификации и другую информацию, связанную с авторизацией.

Взаимодействия:

* Client Machine (Клиентская машина)
  + HTTP Requests (HTTP-запросы): Браузер отправляет HTTP-запросы к веб-серверу для получения и отправки данных.
  + Authentication Requests (Запросы на аутентификацию): Браузер отправляет запросы на аутентификацию к серверу авторизации для входа в систему и проверки токенов.
* Web Server (Веб-сервер)
  + SQL Queries (SQL-запросы): Сервер приложений взаимодействует с базами данных (DoctorDB и UserDB) для чтения и записи данных.
  + Validate Token (Валидация токена): Сервер приложений взаимодействует с сервером авторизации для проверки токенов доступа.
* Auth Server (Сервер авторизации)
  + SQL Queries (SQL-запросы): Сервер авторизации взаимодействует с базой данных авторизации (AuthDB) для хранения и проверки учетных данных и токенов.

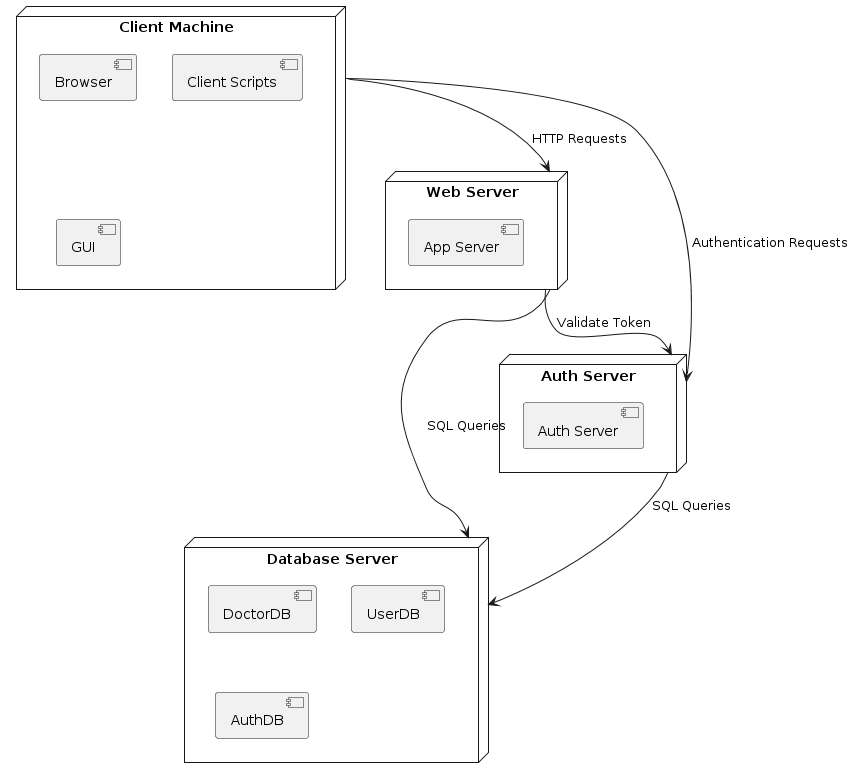


Рисунок 1.3.6.2 – Диаграмма развертывания

1.3.7 Разработка прототипа ИС (Графическое представление)

Графическое представление было разработано в онлайн-сервисе для разработки интерфейсов - *Figma*. Готовый дизайн представлен на рисунках 1.3.7.1, 1.3.7.2 и 1.3.7.3.

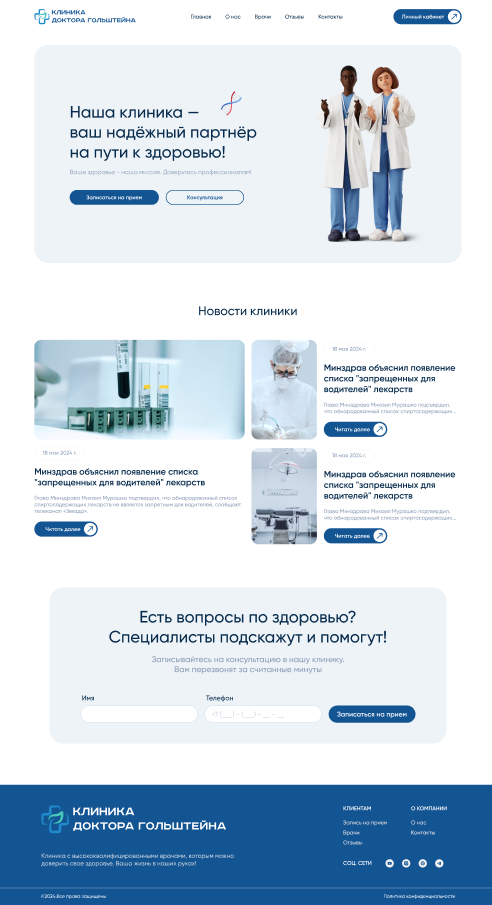


Рисунок 1.3.7.1 – Главная страница

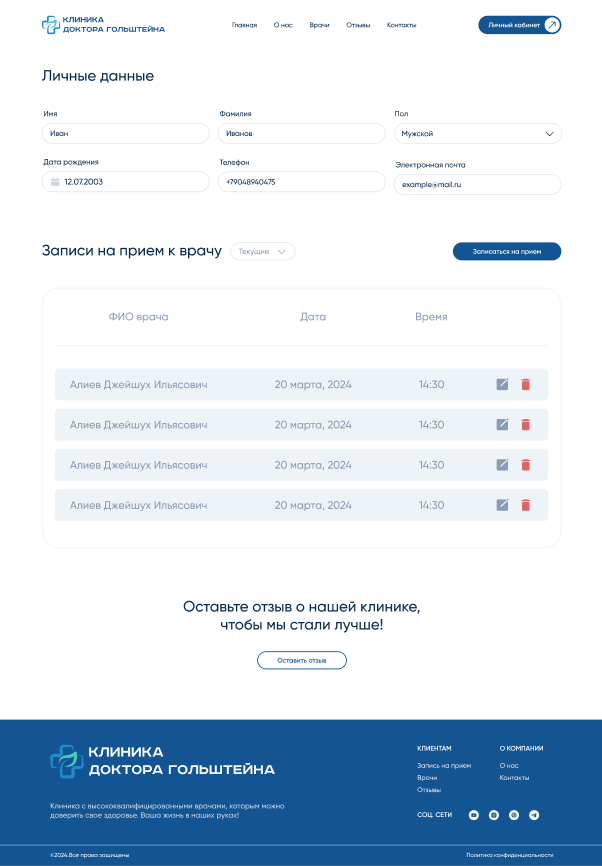


Рисунок 1.3.7.2 – Личный кабинет пользователя

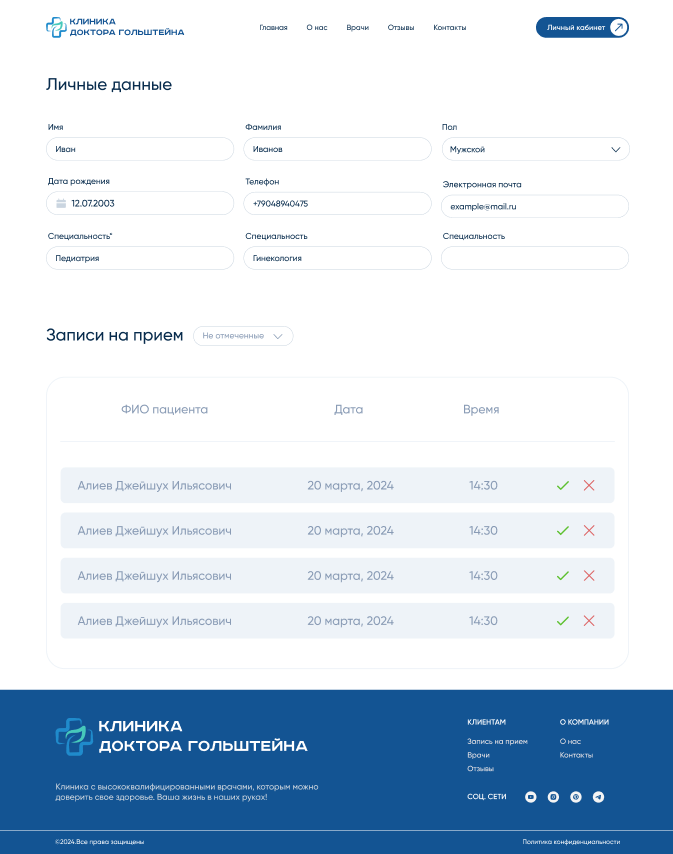


Рисунок 1.3.7.3 – Личный кабинет доктора

## 1.4 Выбор целевой СУБД

В качестве целевой СУБД была выбрана PostgreSQL, которая представляет собой объектно-реляционную базу данных. Данная СУБД обеспечивает высокий уровень надежности и производительности, а также поддерживает сложные запросы и транзакции. PostgreSQL имеет богатый набор функций, таких как поддержка хранимых процедур, триггеров и индексов, что позволяет эффективно управлять данными. Она имеет хорошую интеграцию с Go (Golang), что упрощает создание высокопроизводительных и масштабируемых приложений.

## 1.5 Разработка логической модели предметной области

Сперва выделяем основные сущности, которые были описаны в концептуальной модели, добавляя к ним схемы, необходимые для полного описания предметной области и изменения структуры связей, в частности, нужно преобразовать связи 𝑛 × 𝑚 (многие ко многим) в пары 1 × 𝑛 (один ко многим). К каждой сущности добавляем описание атрибутов: название, тип данных, диапазон возможных значений и смысл атрибута в логической модели. В диапазоне значений мы указываем схему, которая описывает определенный шаблон, которому должны соответствовать данные в поле. Результат представлен на таблице 3. Диаграмма изображена на рисунке 1.5.

Таблица 3 – Описание связей между сущностями.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Сущность** | **Атрибут** | **Тип данных** | **Домен** | **Описание атрибута** |
| Пользователь | ID | Числовой | Натуральные числа | Уникальный идентификатор пользователя |
| Имя | Текстовый | Строка | Имя пользователя |
| Электронная почта | Текстовый | Строка | Адрес электронной почты |
| Телефон | Текстовый | Строка | Номер телефона |
| Пароль | Текстовый | Строка | Пароль для аутентификации |
| Верификация | Логический | Boolean | Верифицирован ли пользователь |
| Доктор | ID | Числовой | Натуральные числа | Уникальный идентификатор доктора |
| Имя | Текстовый | Строка | Имя доктора |
| Специализация | Текстовый | Строка | Специализация доктора |
| Опыт работы | Числовой | Положительные числа | Количество лет опыта работы |
| Запись к врачу | ID | Числовой | Натуральные числа | Уникальный идентификатор записи |
| Дата и время | Дата | Дата и время | Дата и время записи |
| ID пользователя | Числовой | Натуральные числа | Идентификатор пользователя |
| ID доктора | Числовой | Натуральные числа | Идентификатор доктора |
| Статус | Текстовый | Строка | Статус записи (активна, завершена и т.д.) |
| Отзывы | ID | Числовой | Натуральные числа | Уникальный идентификатор отзыва |
| ID пользователя | Числовой | Натуральные числа | Идентификатор пользователя |
| ID доктора | Числовой | Натуральные числа | Идентификатор доктора |
| Текст | Текстовый | Строка | Текст отзыва |
| Дата | Дата | Дата | Дата написания отзыва |
| Рейтинг | Числовой | Положительные числа | Рейтинг в отзыве |
| Заявка на звонок | ID | Числовой | Натуральные числа | Уникальный идентификатор заявки |
| email пользователя | Текстовый | Строка | Email пользователя |
| Дата и время | Дата и время | Дата и время | Дата и время заявки |
| Статус | Текстовый | Строка | Статус заявки |
| Верификационные данные | ID пользователя | Числовой | Натуральные числа | Идентификатор пользователя |
| email | Текстовый | Строка | Email регистрирующегося пользователя |
| Код | Числовой | Положительные числа | Код для верификации пользователя |
| Истекает | Дата и время | Дата и время | Дата и время истечения кода верификации |
| Новости | ID | Числовой | Натуральные числа | Уникальный идентификатор новости |
| ID доктора | Числовой | Натуральные числа | Идентификатор доктора |
| Заголовок | Текстовый | Строка | Заголовок новости |
| Краткий текст | Текстовый | Строка | Краткий текст новости |
| Полный текст | Текстовый | Строка | Полный текст новости |
| Дата публикации | Дата | Дата | Дата публикации новости |

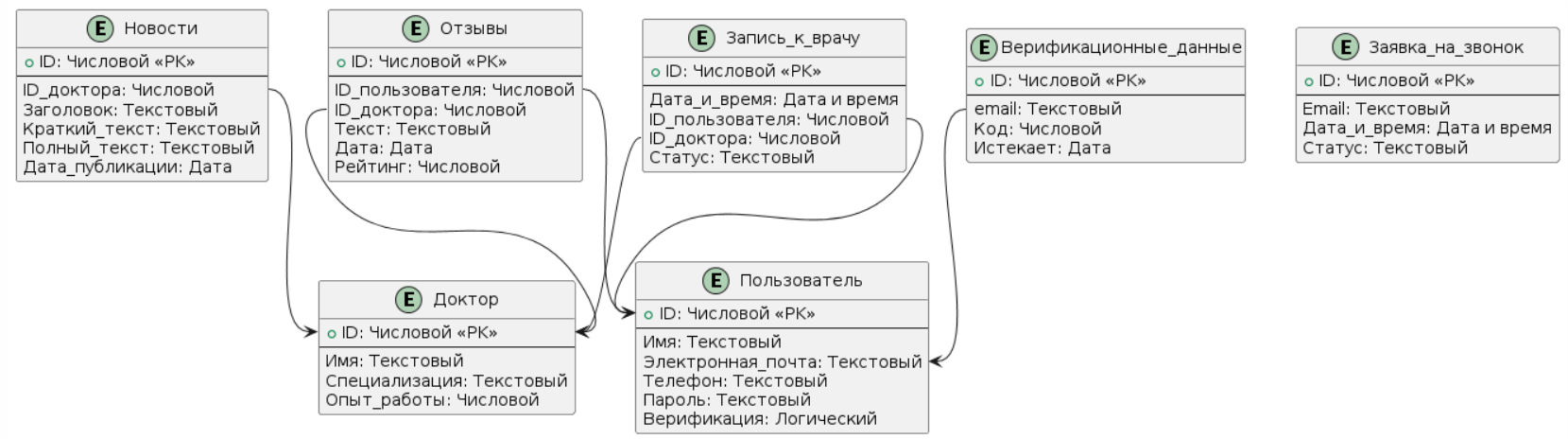


Рисунок 1.5 – ER-диаграмма логической модели.

**1.6 Разработка физической модели предметной области**

В качестве СУБД была выбрана PostgreSQL. Таблицы будут представлять сущности системы, а также предоставлять такую служебную информацию для работы ИС, как аутентификационные токены, сокеты соединения и коды подтверждения личности. Типы полей соответствуют встроенным типам выбранной СУБД. Указаны ограничения на типы ключей, уникальность и другое. В значениях по умолчанию используется встроенный функционал PostgreSQL, например, current\_timestamp и gen\_random\_uuid – текущее время и генерация рандомного UUID соответственно. В содержании указаны примеры данных, которые соответствуют типам и ограничениям полей. Результат представлен в таблице 4.

Таблица 4 – Структура таблиц базы данных.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Наименование**  **поля** | **Тип**  **поля** | **Условие на значение** | **Ключевое поле / Уникальное** | **Значение по умолчанию** | **Содержание** |
| Users (Пользователи) | | | | | |
| id | UUID | Not Null | Primary key | gen\_random\_uuid | 8f873e0b-f39c-4cb7-8b53-f681934f5cfc |
| email | TEXT | Not Null | Unique |  | test1@mail.ru |
| password | TEXT | Not Null |  |  | хэш пароля |
| name | TEXT | Not Null |  |  | Иван |
| surname | TEXT | Not Null |  |  | Курочкин |
| sex | TEXT | Not Null, IN (‘Мужчина’, ‘Женщина’) |  |  | Мужчина |
| birth\_date | DATE | Not Null |  |  | 2004-11-17 |
| phone\_number | TEXT | Not Null |  |  | +7(962)069-49-63 |
| created\_at | DATE | Not Null |  |  | 2024-05-25 |
| is\_verified | BOOLEAN | Not Null |  | false | true |
| Doctor (Доктор) | | | | | |
| id | SERIAL | Not Null | Primary key |  | 1 |
| fullname | TEXT | Not Null |  |  | Алиев Джейхун Ильясович |
| speciality | TEXT | Not Null |  |  | Хирург |
| description | TEXT | Not Null |  |  | Описание доктора |
| medical\_degree | TEXT | Not Null |  |  | Доктор Медицинских Наук |
| work\_experience | INT | Not Null |  |  | 5 |
| is\_head\_doctor | BOOLEAN | Not Null |  | false | false |
| users\_verification\_data | | | | | |
| email | TEXT | Not Null | Primary key |  | test1@mail.ru |
| code | INT | Not Null |  |  | 2312 |
| expires\_at | TIMESTAMPTZ | Not Null |  |  |  |
| Appointment (Запись на прием) | | | | | |
| id | Serial | Not Null | Primary key |  | 1 |
| patient\_uuid | UUID |  | Foreign key |  | 8f873e0b-f39c-4cb7-8b53-f681934f5cfc |
| doctor\_id | INT | Not Null | Foreign key |  | 1 |
| begins\_at | TIMESTAMPTZ | Not Null |  |  | 2024-06-15 10:00:00 |
| ends\_at | TIMESTAMPTZ | Not Null |  |  | 2024-06-15 10:30:00 |
| is\_available | BOOLEAN | Not Null |  | true | false |
| is\_completed | BOOLEAN | Not Null |  | false | false |
| is\_expired | BOOLEAN | Not Null |  | false | false |
| News (Новости) | | | | | |
| id | Serial | Not Null | Primary key |  | 1 |
| title | TEXT | Not Null |  |  | Минимизируем риск невралгии |
| short\_body | TEXT | Not Null |  |  | Короткий текст новости |
| full\_body | TEXT | Not Null |  |  | Полный текст новости |
| publication\_date | DATE | Not Null |  |  | 2024-05-15 |
| doctor\_id | INT | Not Null | Foreign key |  | 2 |
| Review | | | | | |
| id | Serial | Not Null | Primary key |  | 1 |
| rating | INT | Not Null,  0< rating <=5 |  |  | 5 |
| patient\_uuid | UUID | Not Null | Foreign key |  | 8f873e0b-f39c-4cb7-8b53-f681934f5cfc |
| doctor\_id | INT | Not Null | Foreign key |  | 1 |
| body | TEXT | Not Null |  |  | Текст отзыва |
| publication\_date | DATE | Not Null |  |  | 2024-05-14 |
| Call\_request (Запрос на звонок) | | | | | |
| id | Serial | Not Null | Primary key |  | 1 |
| name | TEXT | Not Null |  |  | Алексей |
| phone | TEXT | Not Null |  |  | +7(912)345-67-89 |
| request\_time | TIMESTAMPTZ | Not Null |  | current\_timestamp | 2024-06-10 14:30:00 |
| status | TEXT | Not Null, IN ('pending', 'completed', 'canceled') |  | 'pending' | pending |

Перенесем таблицу 4 в ER-диаграмму в нотации Чена. Результат представлен на рисунке 1.6.1.

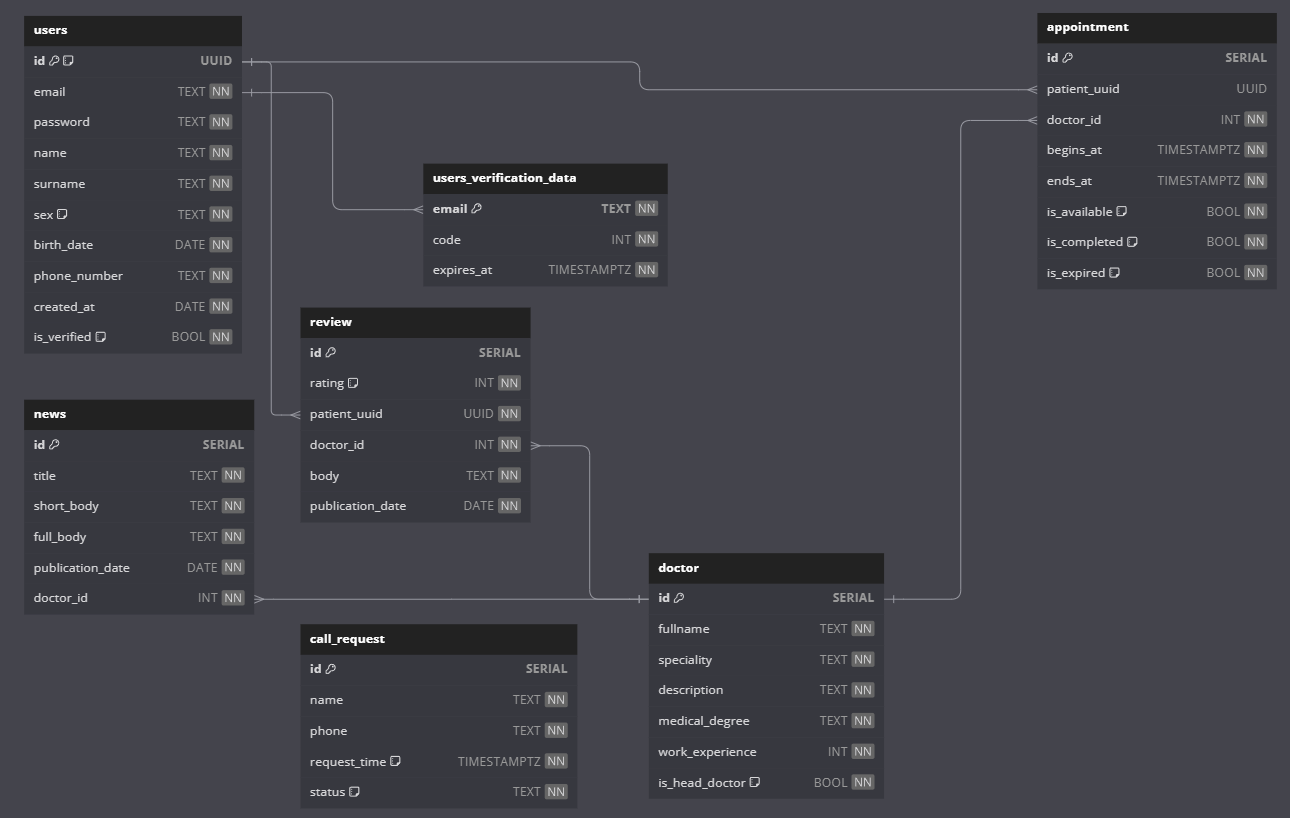


Рисунок .6.1 – ER-диаграмма dbdiagram.io физической модели.

Для построения был использован веб-сайт dbdiagram.io. На нем можно удобно посмотреть связи в таблице и ограничения при наведении на соответствующие атрибуты, пример представлен на рисунке 1.6.2 и 1.6.3.

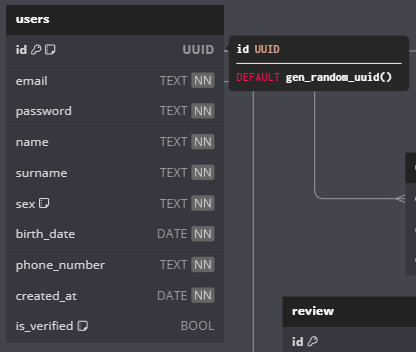


Рисунок 1.6.2 – Пример просмотра поля по умолчанию

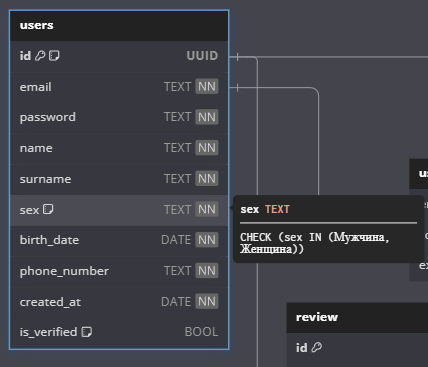


Рисунок 1.6.3 – Пример просмотра ограничения на поле

# 2 Разработка программного продукта

## 2.1 Структура программного продукта

Структура программы состоит из двух частей: серверная и клиентская. Серверная часть отвечает за хранение и работу с бизнес логикой, за предоставление клиентам необходимых ресурсов и взаимодействие с БД и файловой системой. Серверная часть предоставляет интерфейс прикладного программирования (Application Programming Interface - *API)* для работы с ней, которое организовано и спроектировано в *REST (*представление состояния передачи *(Representational State Transfer)* стиле. Сервер состоит из четырех модулей: маршруты, контроллеры, сервисы и слой доступа к данным (Data Access Layer – *DAL)*. В модуле маршрутов устанавливаются контроллеры на определенные пути, по которым идет обращение к серверу. Модуль контроллеров содержит логику обработки запросов и формирования конечных ответов с их заголовками и статусом ответа, а также перехвата ошибок. При получении запроса информация о нем передается в контроллер, который вызывает сервисы, в которых содержится вся бизнес логика приложения. В сервисах описывается преобразование структуры сущностей, их форматирование и работа с ними. Для фиксирования проделанной работы в сервисах используется модуль *DAL*, который взаимодействует с базой данных. После фиксирования работы с БД данные ответа спускаются вниз, проходя все модули. Общая схема архитектуры представлена на рисунке 2.1.

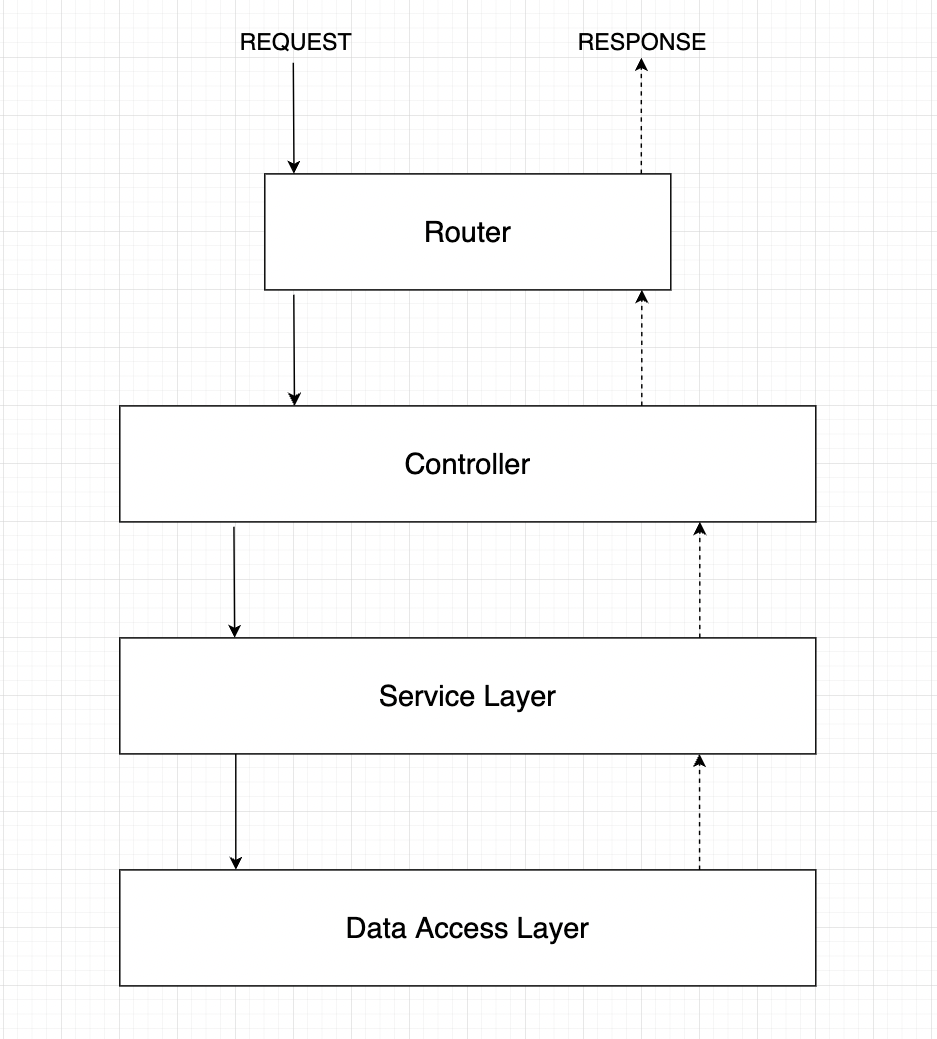


Рисунок 2.1 - Общая схема архитектуры

Клиентская часть отвечает за логику работы с пользователем, клиентскую маршрутизацию по странице и предоставление интерфейсов. Клиентская часть реализована с помощью технологии *SPA* (одностраничное приложение *(Single Page Application)*), которая позволяет динамически обновлять содержимое страницы без полной перезагрузки, что увеличивает качество пользовательского опыта, *FCP* (отрисовка первого контента (*First Contentful Paint*) и *TTI* (время до интерактивности *(Time To Interactive)*). Для рендеринга страниц используется технология *CSR (*клиентская отрисовка *(Client-Side Rendering)*), которая позволяет отрисовывать страницу на стороне браузера, благодаря чему приложение меньше весит и его проще разрабатывать. Клиентское приложение построено с помощью модульной архитектуры, которая предоставляет однонаправленный поток данных. В данной системе существует четыре части: *UI*, компоненты, модули и страницы. В части *UI* (пользовательский интерфейс (*User Interface*)) хранятся переиспользуемые блоки кода, которые не содержат бизнес логику, например, модальные окна, кнопки, поля ввода и так далее. В компонентах хранятся более сложные блоки, которые имеют небольшую часть бизнес логики, но без контекста их использование будет затруднительно. В части модулей хранятся сложные и независимые части приложения, которые содержат основную логику работы с пользователем. К этой части можно отнести форму регистрации. На уровне страниц происходит комбинация модулей, а также описание расположения основных частей страницы, таких как шапка или футор. На каждом из уровней можно использовать блоки частей, которые находятся ниже по уровню, например, в модулях можно использовать *UI* и компоненты, но страницы и сами модули в других модулях использовать нельзя. Благодаря данной архитектуры код становится переиспользуемым, он имеет однонаправленный поток данных (Страницы – Модули – Компоненты – *UI*), а также намного проще вносить изменения и удалять лишние части. Связи между частями прозрачны, каждый блок имеет свою зону ответственности. Клиентская часть предоставляет защищенный доступ к определенным маршрутам. Так, для перехода в личный аккаунт, пользователю нужно быть авторизованным в системе.

Клиентская и серверная часть взаимодействуют с помощью *REST* *API*, основанном на стандартных *HTTP*-методах (протокол передачи гипертекста (Hypertext Transfer Protocol)) и ресурсоориентированном подходе. В качестве формата данных был выбран *JSON*, так как он легок и компактен, а также у него хорошая поддержка в браузере. Для поддержания *RTC* (все пользователи могут обмениваться информацией мгновенно или с незначительной задержкой или задержкой передачи) используется технология *WebSocket*, который предоставляет двунаправленный протокол связи между клиентом (браузером) и сервером, позволяющий обмениваться сообщениями в режиме реального времени. Он устанавливает одно соединение и передает ответ на единственный запрос в тот момент, когда ответ появился — без дополнительных запросов, как у *HTTP*-протокола. Запросы и ответы приходят без задержек и сетевой нагрузки.

## 2.2 Реализация бизнес правил

В качестве основных бизнес-правил будет описан процесс авторизации и записи к врачу.

2.2.1 Авторизация

В качестве данных для авторизации пользователя требуется электронная почта и пароль. Если вводимые пользователем данные соответствуют схеме валидации электронной почты и пароля, то они отправляются на сервер по маршруту */auth/login*. На сервере маршрутизатор перенаправляет запрос в контроллер, который разбирает тело запроса, преобразуя его в *JavaScript* объект. После он передается в слой сервисов, где пароль хешируется (преобразуется в хеш), и все данные передаются в слой *DAL*, где происходит сравнение на соответствие почты и пароля. Если соответствие успешно, то возвращается основная информация о пользователе, его имя, фамилия и название файла изображения профиля. После на уровне сервисов, исходя из названия файла изображения, создается путь до этого файла и весь отформатированный объект передается на уровень контроллера. В нем создается код ответа и заголовки, объект преобразуется в *JSON* и отправляется клиенту. Если соответствие прошло неуспешно, то возвращается ошибка об отмене неправильно введенных электронной почты или пароля. После этого на клиенте, при получении успешного ответа, *JSON* формат преобразуется в *JavaScript* объект и происходит переадресация на главную страницу, где создается *WebSocket* соединение с сервером.

2.2.2 Запись на прием

При записи на прием к врачу клиент отправляет серверу запрос на адрес */appointments/create-new-appointment/:appointmentID,* а также передает параметр appointmentID и заголовок авторизации с jwt-токеном. После чего сервер посылает запрос к базе данных, где запись к врачу становится “занятой” и ей присваивается определенный пациент. После этого сервер получает недавно созданную запись из базы данных и отправляет ее на клиент посредством объекта JSON.

Листинг 1 – Код реализации создания записи к врачу на сервере

|  |
| --- |
| func (h \*handler) makeNewAppointment(w http.ResponseWriter, r \*http.Request) {      w.Header().Set("Content-Type", "application/json")      uuid := r.Header.Get("user\_id")      params := httprouter.ParamsFromContext(r.Context())      appointmentID := params.ByName("appointmentID")      id, err := strconv.Atoi(appointmentID)      if err != nil {          http.Error(w, err.Error(), http.StatusInternalServerError)          return      }      err = h.repository.MakeAppointmentByUUID(id, uuid)      if err != nil {          http.Error(w, err.Error(), http.StatusInternalServerError)          return      }      appointment, err := h.repository.GetAppointmentByID(id)      if err != nil {          http.Error(w, err.Error(), http.StatusInternalServerError)          return      }      data, err := json.Marshal(appointment)      if err != nil {          http.Error(w, err.Error(), http.StatusBadRequest)          return      }      w.Write(data)      w.WriteHeader(http.StatusAccepted)  } |

## 2.3 Руководство пользователя

2.3.1 Назначение программы: информационная система предоставляет возможность записи к врачу, просмотра новостей больницы, оставления отзыва о приеме, а также полное управление записью: изменение, отмена.

2.3.2 Выполнение программы:

При входе в систему пользователь попадает на главную страницу, которая изображена на рисунке 2.3.1.

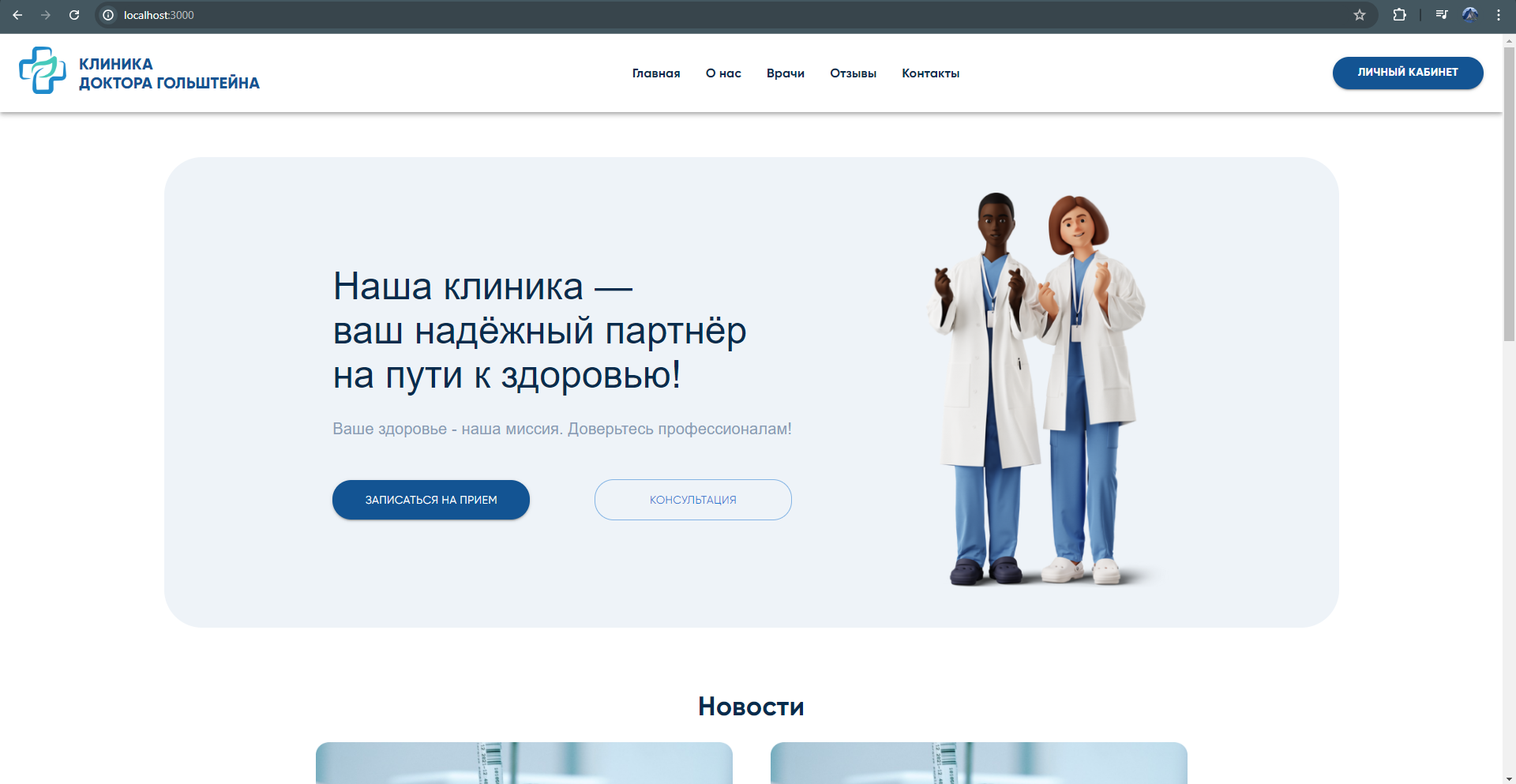


Рисунок 2.3.1 – Главная страница сайта

Если пользователь промотает страницу вниз, то перед ним появится блок с новостями (карусель). Здесь пользователь может ознакомиться с новостями больницы. Блок отображен на рисунке 2.3.2.

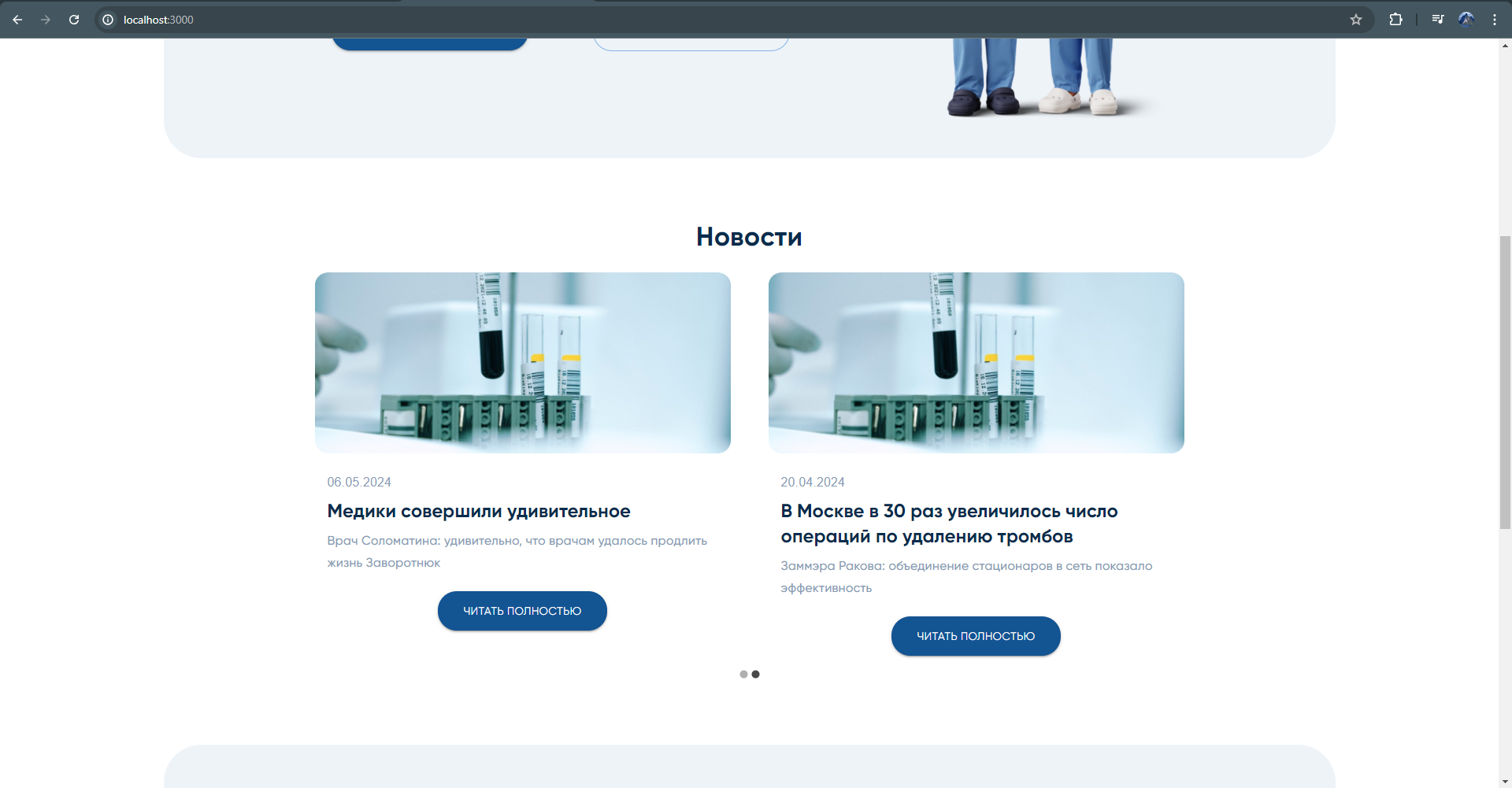


Рисунок 2.3.2 – Блок новостей на сайте

Если пользователь проскролит еще ниже, то перед ним предстанет блок для оставления контактных данных для обратного звонка, а также футор с информацией для пользователя, например, социальные сети больницы, а также запись на прием, список врачей, а также отзывы. Форма отображена на рисунке 2.3.3.

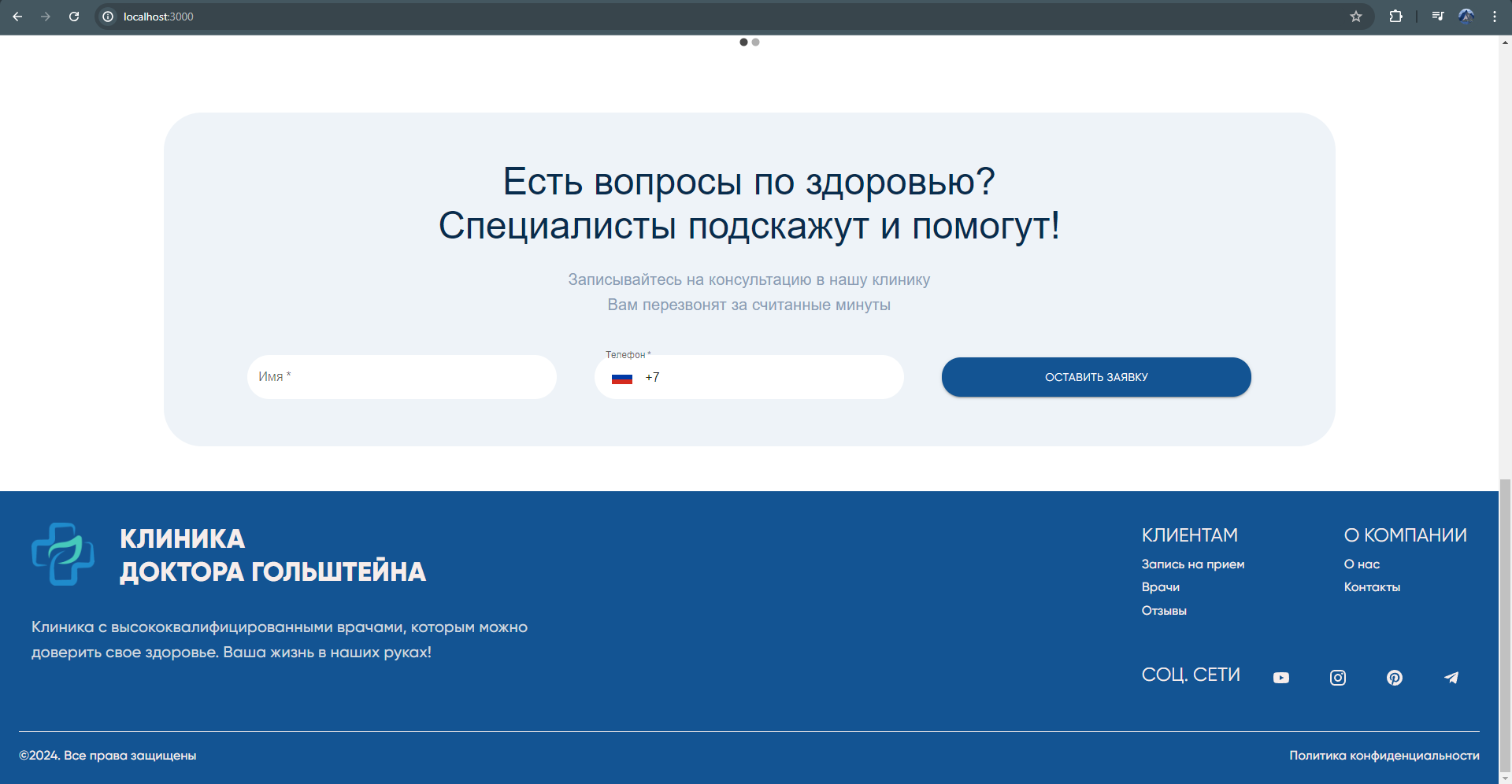


Рисунок 2.3.3 – Форма для оставления контактных данных

В случае, если пользователь нажмет на кнопку “Запись на прием” или “Личный кабинет”, то его перенаправляет на страницу авторизации. Если у него есть аккаунт, то он может ввести данные для авторизации, а именно электронную почту и пароль. Если у пользователя нет аккаунта, то ему нужно нажать на «Нет аккаунта? Зарегистрироваться». Форма входа отображена на рисунке 2.3.4. Форма регистрации изображена на рисунке 2.3.5.

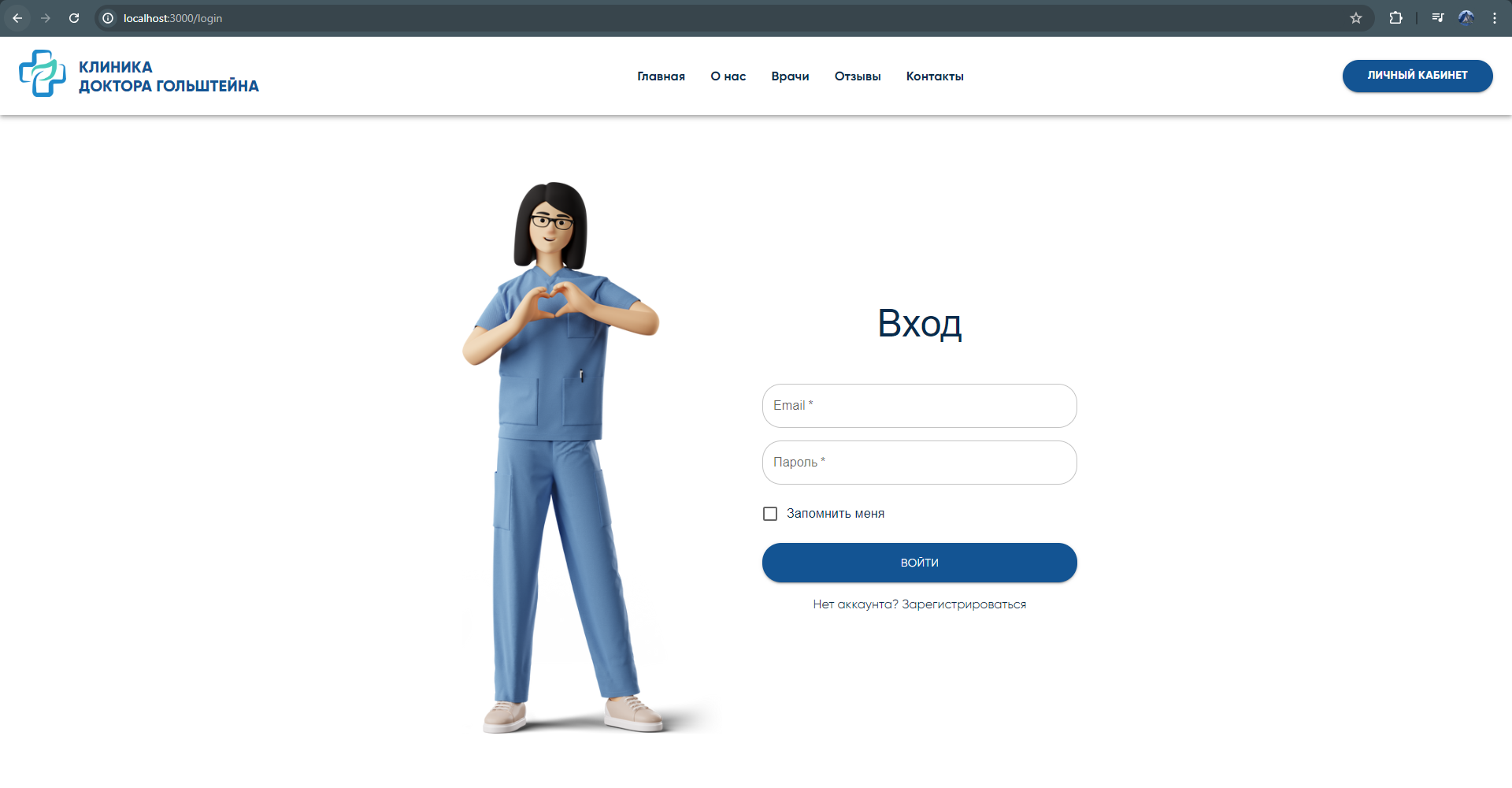


Рисунок 2.3.4 – Форма входа в системе

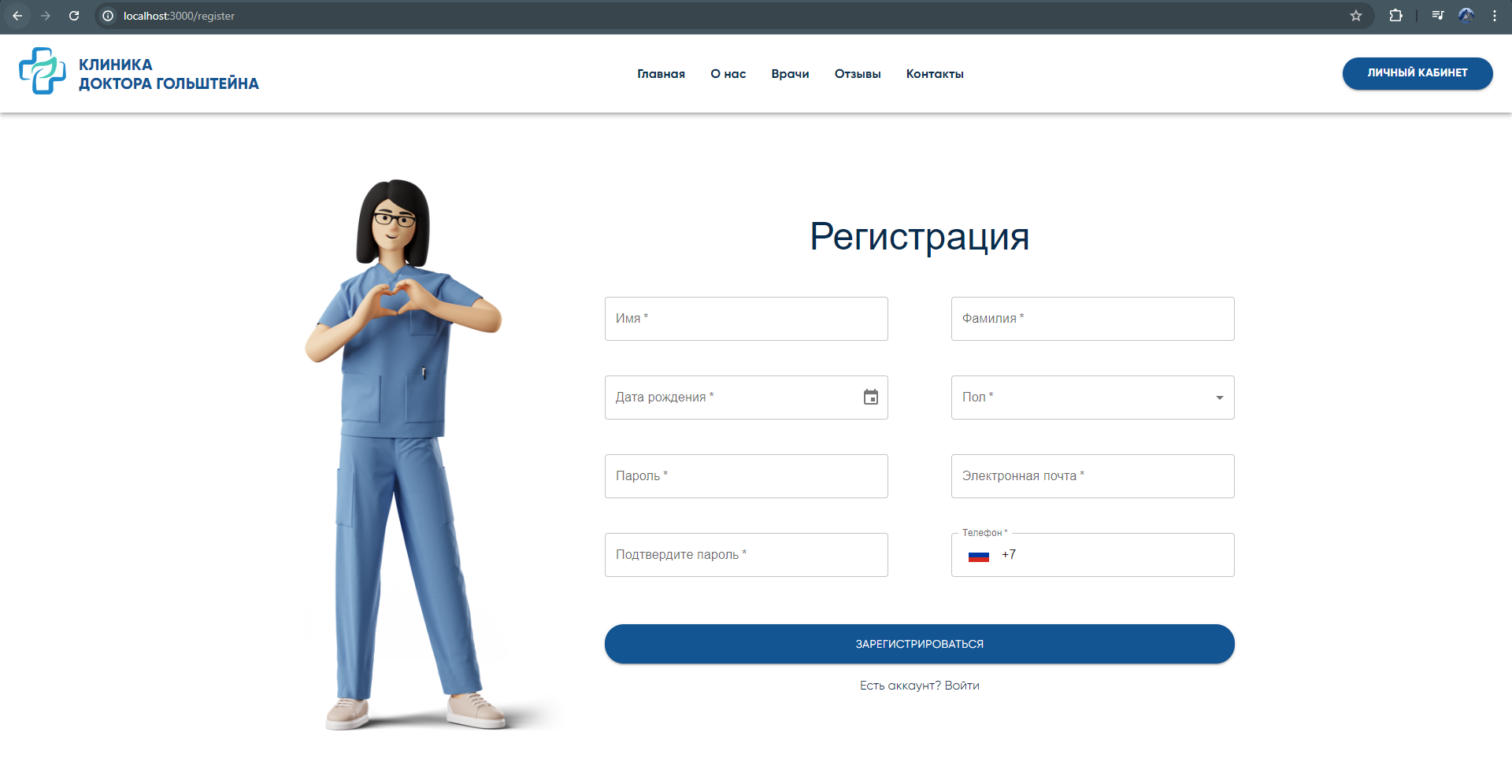


Рисунок 2.3.5 – Форма входа в системе

После ввода данных в форму регистрации (рисунок 2.3.6), в случае валидности введенных данных, пользователь перенаправляется на страницу подтверждения email (рисунок 2.3.7), вместе с этим пользователю на почту приходит email с кодом подтверждения (рисунок 2.3.8). После ввода кода подтверждения пользователь оповещается об успешном создании аккаунта (рисунок 2.3.9) и перенаправляется на страницу авторизации.

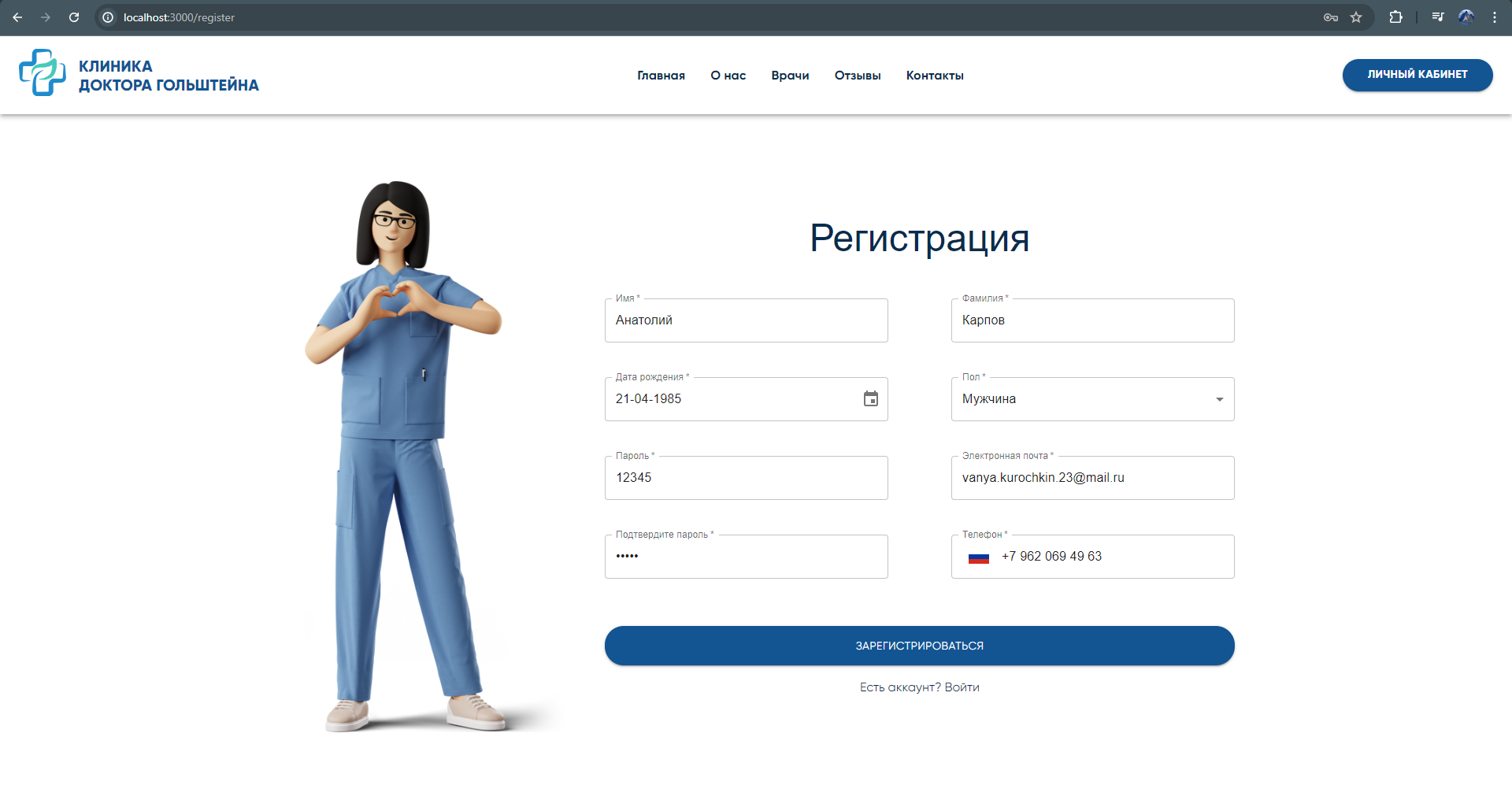


Рисунок 2.3.6 – Заполненная форма регистрации

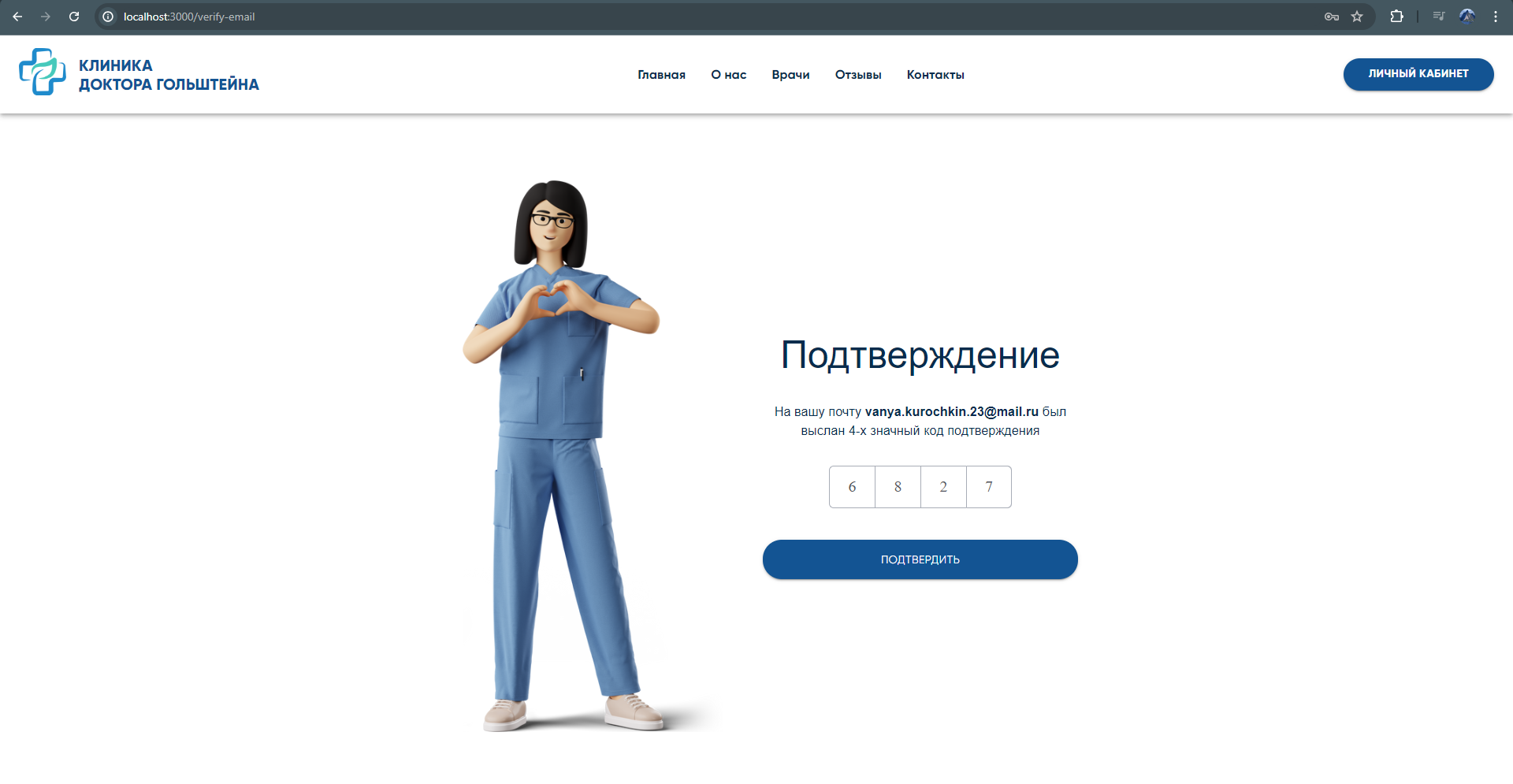


Рисунок 2.3.7 – Страница с кодом подтверждения

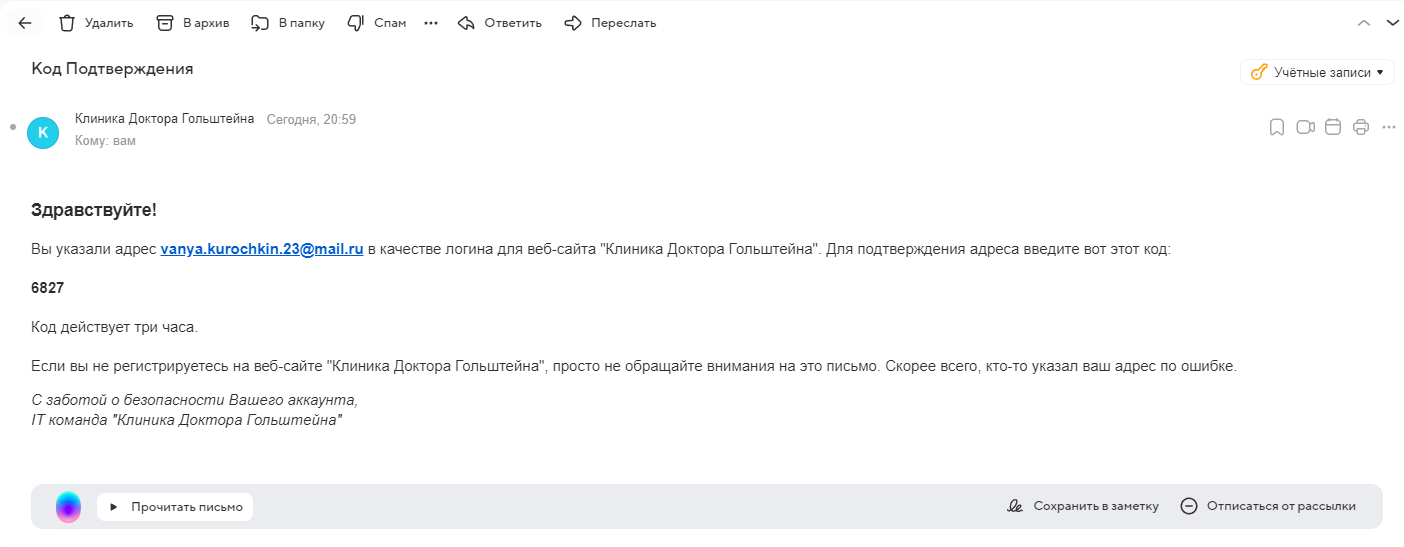


Рисунок 2.3.8 – Email с кодом подтверждения

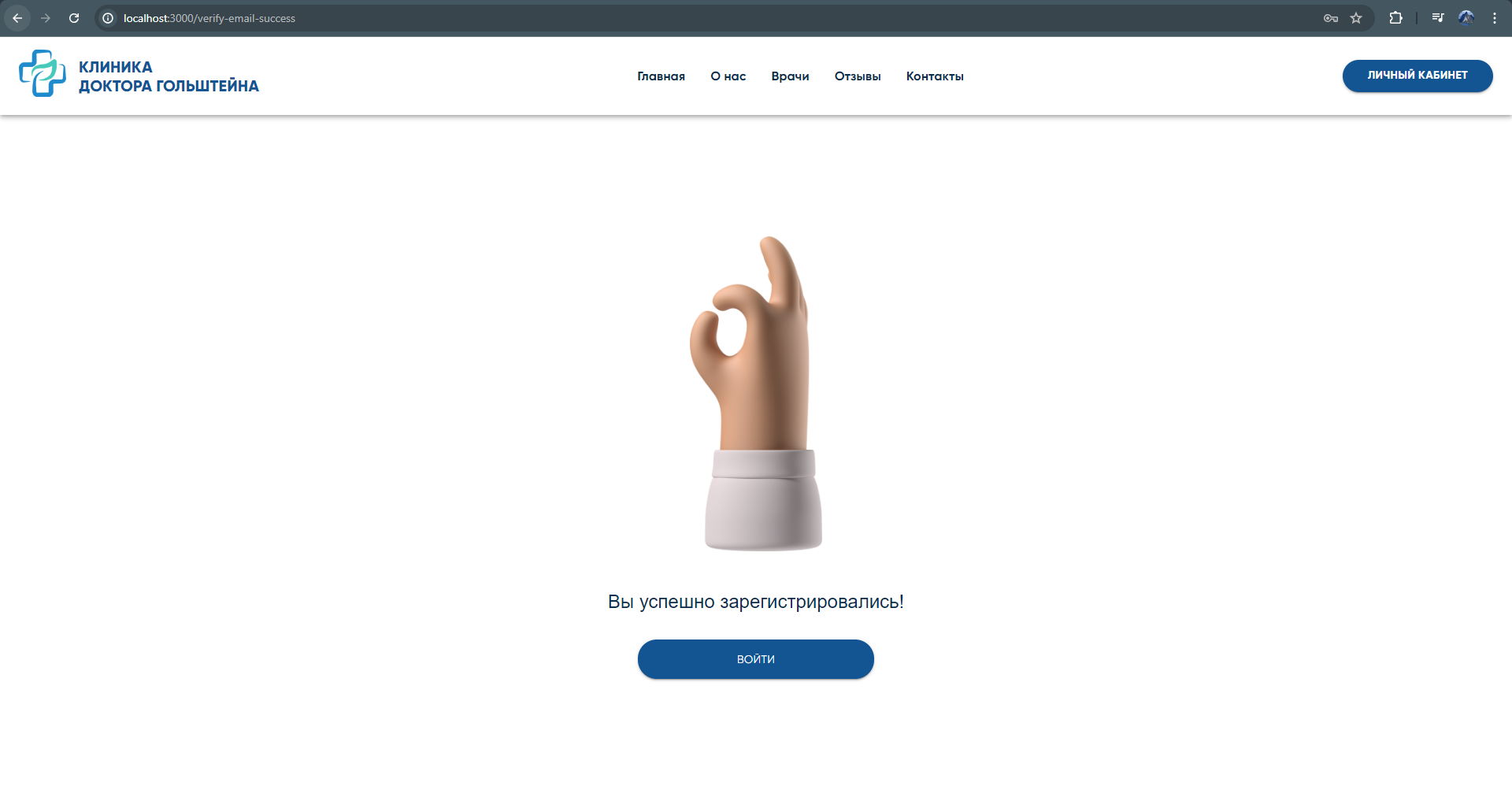


Рисунок 2.3.9 – Оповещение об успешной регистрации

После ввода данных для авторизации на странице для входа, пользователь может записаться на прием и зайти в личный кабинет. В личном кабинете пользователь может увидеть данные, введенные при регистрации, просмотреть все свои записи (текущие и архивные), изменить, а также отменить записи. Личный кабинет отображен на рисунке 2.3.10 и 2.3.11.

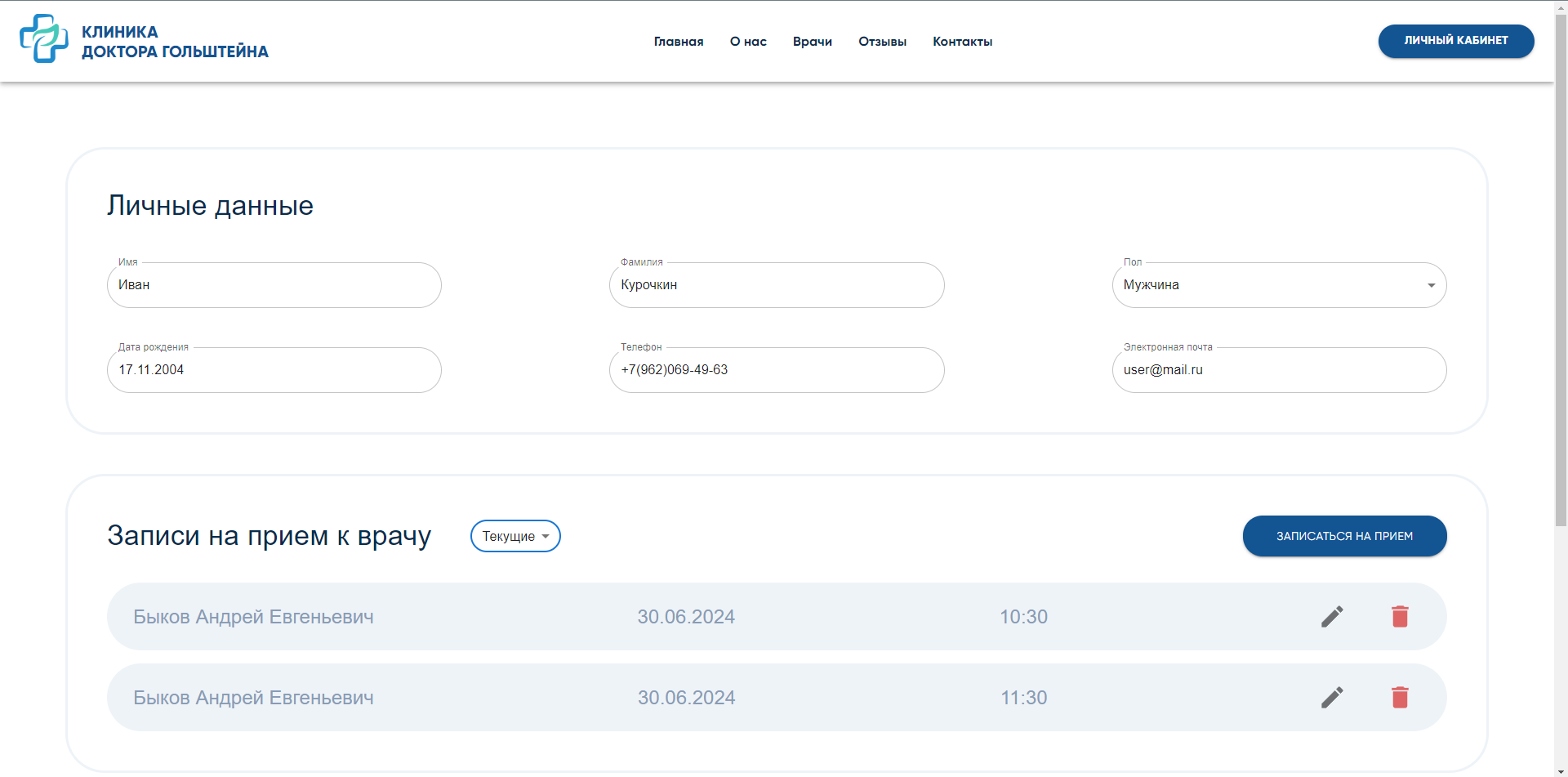


Рисунок 2.3.10 – Личный кабинет (текущие)



Рисунок 2.3.11 – Личный кабинет (архивные)

Главный прецедент “Запись к врачу” активируется при нажатии на кнопку “Записаться на прием” (из личного кабинета или с главной страницы). Сперва пользователь выбирает специальность врача, после ФИО врача, после этого перед пользователем всплывает календарь с датами. После выбора даты появляется возможное время для записи. После выбора времени записи появляется кнопка “Записаться к врачу”. После нажатия на кнопку, запись добавляется в личный кабинет пользователя. Форма для записи к врачу отображена на рисунке 2.3.12.

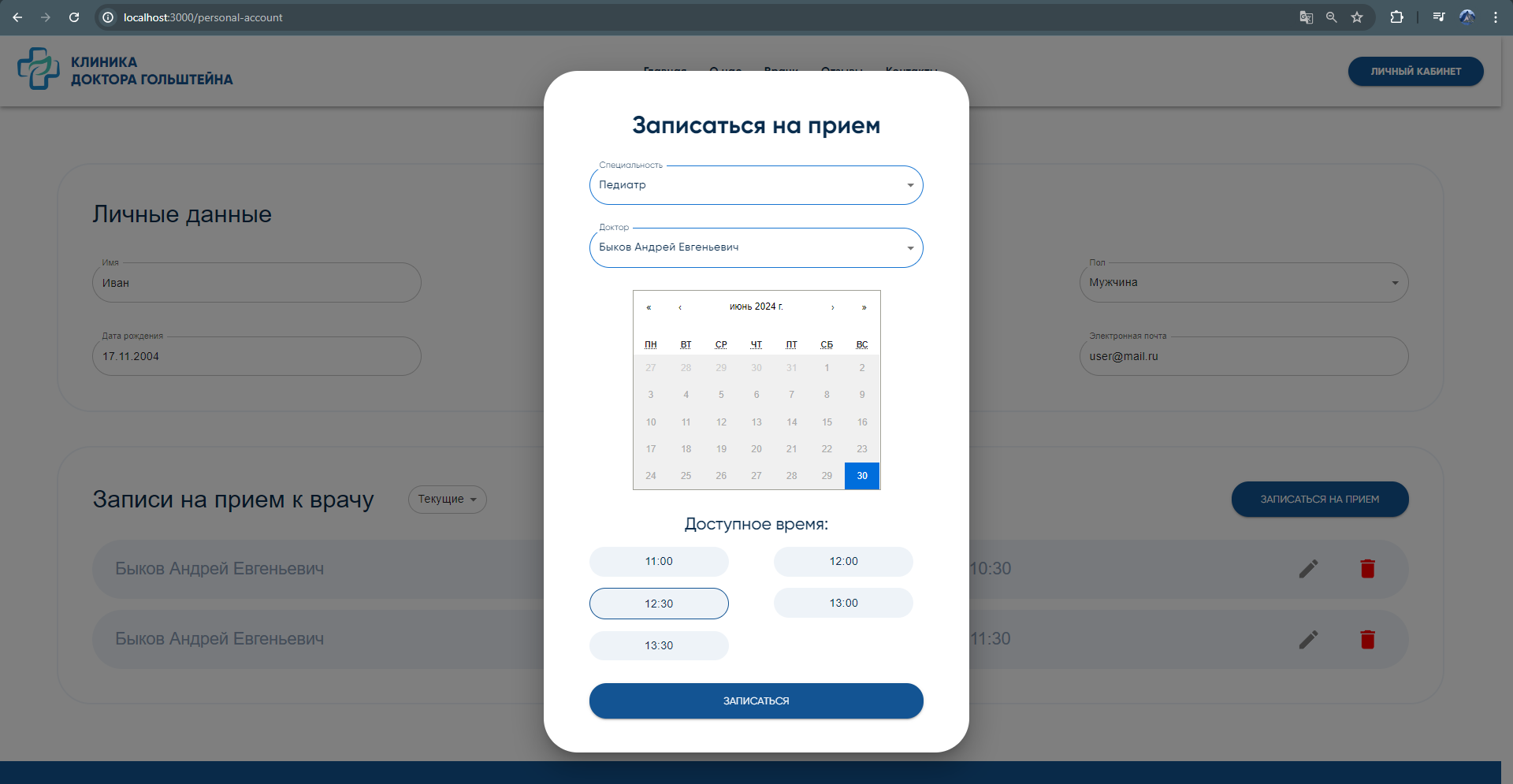


Рисунок 2.3.12 – Форма записи к врачу

Также, можно отредактировать запись к врачу, изменив дату/время записи, что отображено на рисунке 2.3.13, и отменить запись к врачу, что отображено на рисунке 2.3.14.

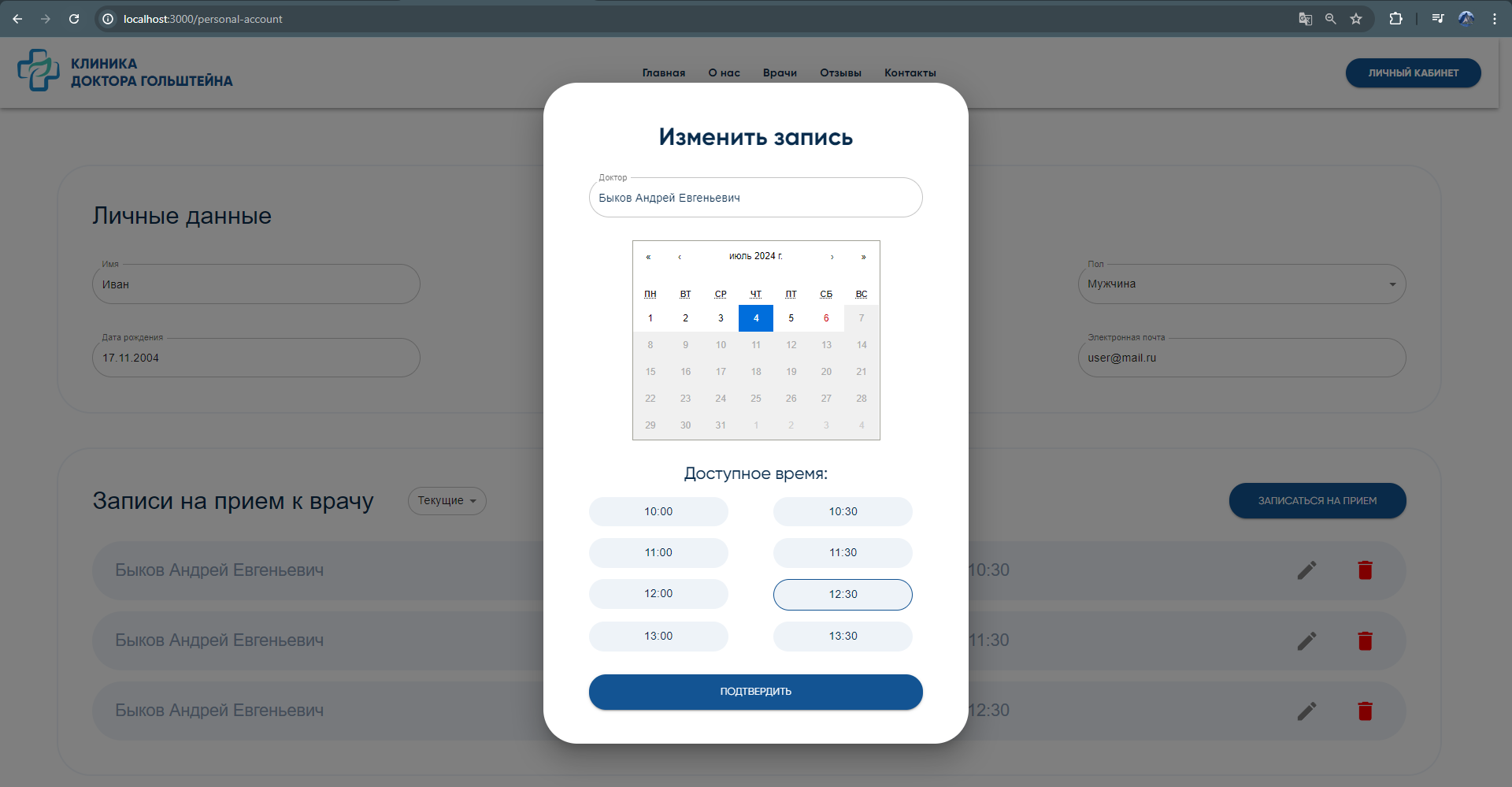


Рисунок 2.3.13 – Изменение записи к врачу

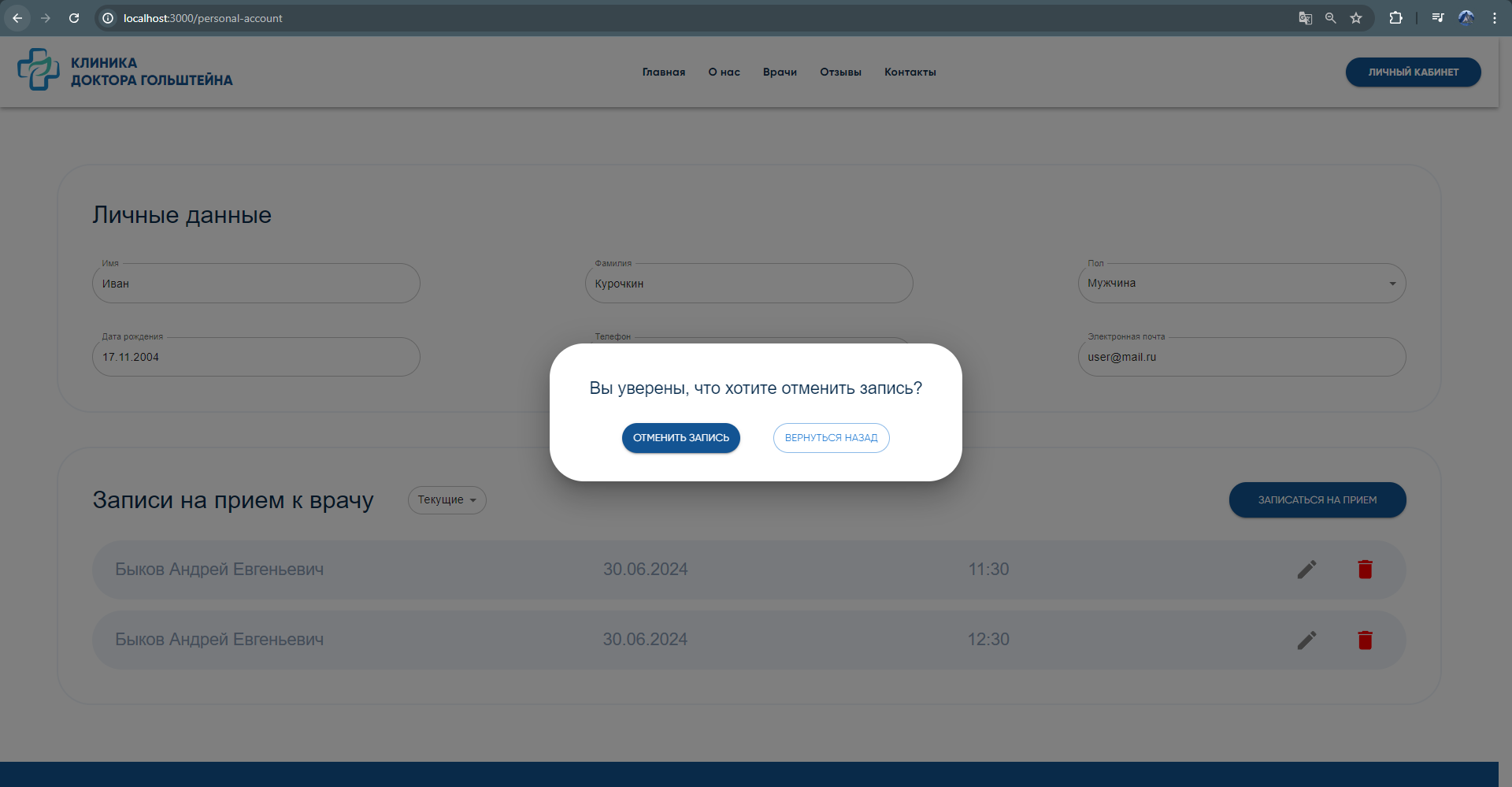


Рисунок 2.3.14 – Отмена записи к врачу

## 2.4 Руководство работника

2.4.1 Назначение программы: информационная система предоставляет возможность врачу отмечать факт посещения пациентом записи к врачу.

2.4.2 Выполнение программы:

При авторизации в системе в качестве доктора, работник может перейти в личный кабинет, в котором будут отражены записи к нему (отмеченные и неотмеченные). Также у доктора присутствует возможность отметить факт посещения пациента, при том нельзя отметить посещение записи, которая еще не состоялась. Личный кабинет работника отображен на рисунках 2.4.1 и 2.4.2.

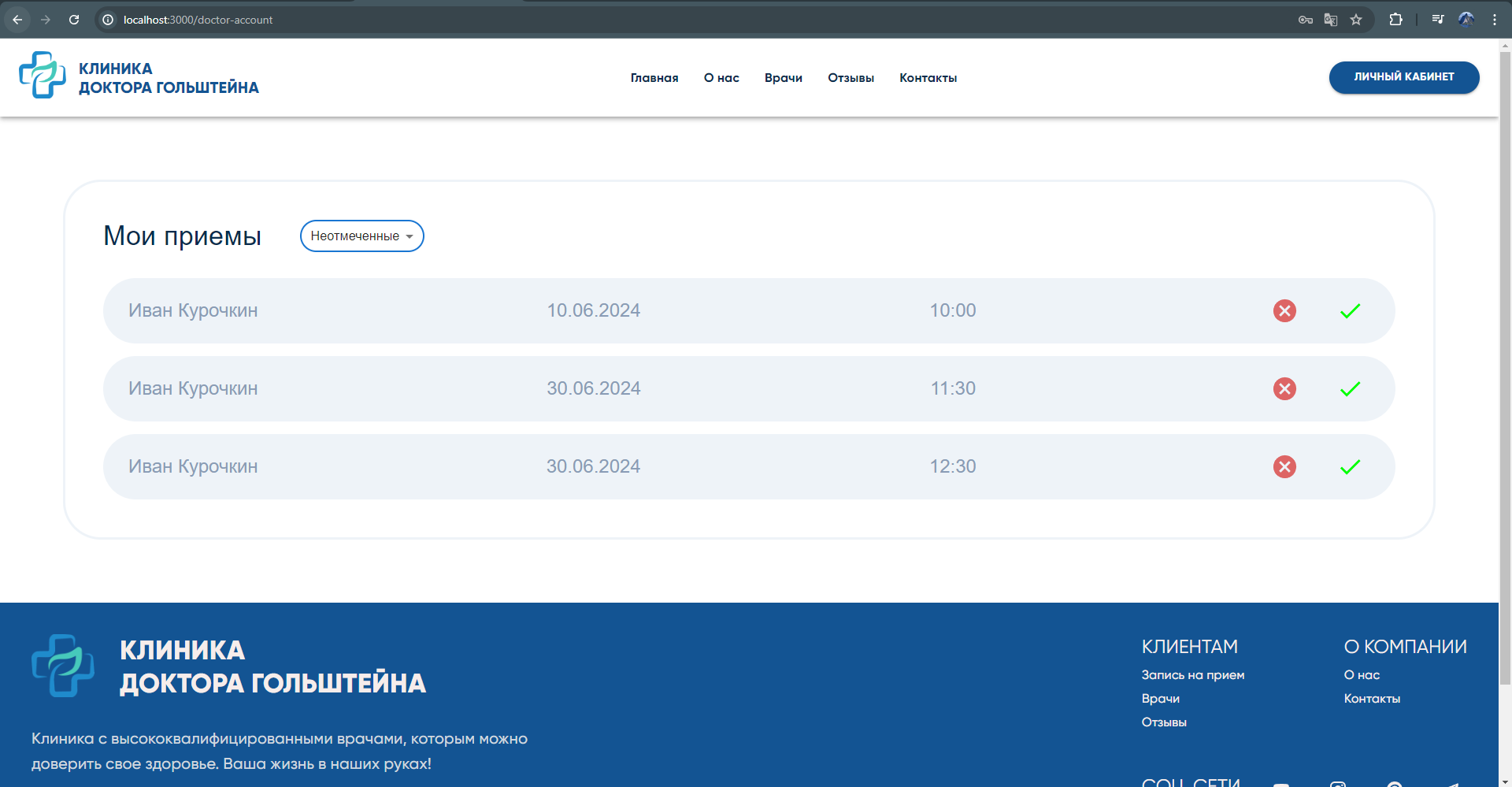


Рисунок 2.4.1 – Личный кабинет доктора (неотмеченные)

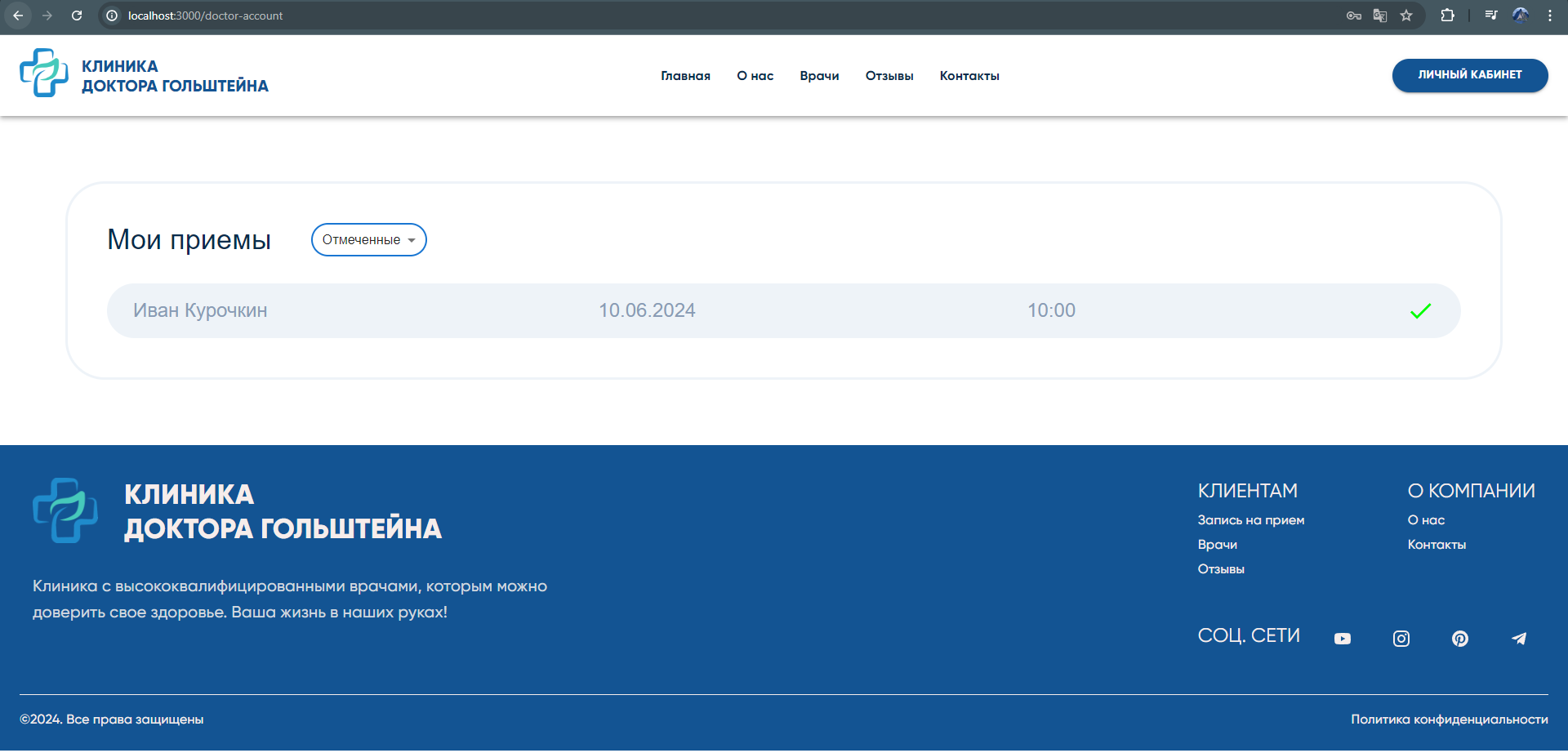


Рисунок 2.4.2 – Личный кабинет доктора (отмеченные)

## 2.5 Тестирование программного продукта

Для проверки соответствия программного обеспечения установленным требованиям было проведено несколько уровней тестирования. Были выполнены как функциональные, так и нефункциональные тесты.

Функциональное тестирование включало unit тесты, которые проверяют работу отдельных функций или частей программы, отвечающих за конкретные задачи; интеграционные тесты, оценивающие правильность взаимодействия модулей между собой; и E2E (сквозное) тестирование, проверяющее работу приложения на всем пользовательском пути. Для unit тестов использовался Jest, который хорошо поддерживается как на серверной, так и на клиентской стороне, позволяя легко заменять используемые зависимости на mock-объекты и работать с сериями тестов. Для интеграционных тестов на клиенте использовался React Testing Library, обеспечивающий проверку целостности различных модулей, а на сервере также применялся Jest. E2E тесты выполнялись с помощью Playwright, предоставляющего удобное API для симуляции пользовательских действий и событий в системе. Дополнительно на клиентской стороне проводились скриншотные тесты с использованием Storybook для проверки соответствия компонентов страницы заявленному интерфейсу.

Для нефункционального тестирования использовался встроенный в devtools браузера инструмент Lighthouse, предназначенный для аудита производительности и доступности веб-ресурса. Этот инструмент позволяет выявить проблемные места сайта, улучшить качество страниц и их работу. С его помощью проводилось нагрузочное тестирование, тестирование безопасности, доступности, а также оптимизация для поисковых систем (SEO).

# ЗАКЛЮЧЕНИЕ

В ходе курсовой работы была спроектирована и разработана автоматизированная информационная система, предоставляющая возможность поликлинике автоматизировать процесс записи к врачу, снять нагрузку с администраторов. Благодаря реализованным диаграммам и схемам, которые рассматривают систему с разных сторон, увеличилась скорость и качество разработанного продукта. Итоговая система соответствует заявленным требованиям к функциональности, производительности и безопасности, что подтверждается проведенным тестированием.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Леоненков, А. В. Самоучитель UML / А. В. Леоненков – 2-е изд., перераб. и доп. – Санкт-Петербург: БХВ – Петербург, 2004 – 312 с.
2. Ларман, К. Применение UML и шаблонов проектирования / К. Ларман – 2-ое издание. Перевод с английского. – Москва: Издательский дом «Вильямс», 2004. – 624 с.
3. Советов, Б.Я. Базы данных: учебник для прикладного бакалавриата/ Б. Я. Советов, В.В. Цехановский, В. Д. Чертовской – 2-е изд. Москва: Издательство Юрайт, 2016 – 463 с.
4. Коннолли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли, К. Бегг – 3-е изд. – Москва: Вильяме, 2003. – 1440с.

# ПРИЛОЖЕНИЕ А

**Техническое задание на разработку**

Разрабатываемая информационная система предоставляет возможность записи к врачу, управления своими записями, просмотра новостей больницы, а также возможность отметки факта посещения пациентом приема к врачу.

Техническое задание

1 Общие сведения

1.1 Наименование системы

Полное: Автоматизированная информационная система “Запись на прием к врачу”

Краткое: АИС «Запись на прием к врачу»

1.2 Краткая характеристика области применения

Приложение предусмотрено для оптимизации работы больницы, управления записями к врачу.

1.3 Плановые сроки работ

07.02 – 09.02 постановка задачи; разработка технического задания;

10.02 – 20.02 техническое проектирование; построение диаграмм;

21.02 – 26.02 разработка структуры БД; заполнение таблиц, приведение к 3НФ;

27.02 – 05.03 разработка структуры программы, входных и выходных данных;

06.03 – 30.04 программирование; отладка программы;

01.05 – 19.05 разработка программной документации;

20.05 – 25.05 оформление пояснительной записки.

1.4 Условные обозначения и сокращения

3НФ – 3-я нормальная форма

API – интерфейс прикладного программирования (*Application* *Programming* *Interface*)

CSR – клиентская отрисовка (*Client-Side Rendering*)

DAL – слой доступа к данным (*Data Access Layer*)

DOM – объектная модель документа (*Document Object Model*)

E2E – сквозное тестирование (*End-to-End*)

FCP – отрисовка первого контента (*First Contentful Paint*)

GUI – графический пользовательский интерфейс (*Graphical User Interface*)

HTTP – протокол передачи гипертекста (*Hypertext Transfer Protocol*)

ИС – информационная система

JSON – нотация объектов JavaScript (*JavaScript Object Notation*)

ORM – сопоставление объектов и реляционных баз данных (*Object-Relational Mapping*)

REST – представление состояния передачи (*Representational State Transfe*r)

RTC – связь в реальном времени (*Real-Time Communication*)

SEO – оптимизация для поисковых систем (*Search Engine Optimization*)

SPA – одностраничное приложение (*Single Page Application*)

SQL – язык структурированных запросов (*Structured Query Language*)

TTI – время до интерактивности (*Time To Interactive*)

UI – пользовательский интерфейс (*User Interface*)

БД – база данных

СУБД – система управления базами данных

2 Основания для разработки

Основанием для разработки является задание на курсовую работу по дисциплине «Проектирование информационных систем», выданное 06.02.2024 преподавателем Коченовым Дмитрием Александровичем.

2.1 Наименование и условное обозначение темы разработки

Разработка АИС «Запись на прием к врачу».

2.2 Заказчик

Кафедра «Системы искусственного интеллекта» ИКИТ СФУ.

2.3 Разработчики

Студент группы КИ22-21б Курочкин Иван Дмитриевич.

3 Назначение разработки

3.1 Функциональное назначение

Приложение должно обладать следующим функционалом:

- авторизация и аутентификация пользователя;

- просмотр новостей больницы;

- запись к врачу: создание, изменение, удаление;

- оставление контактных данных для обратной связи;

- возможность отметить факт посещения пациентом приема к врачу.

3.2 Цели

Разработка веб-приложения для управления записями к врачу в режиме реального времени, которое предоставляет возможность больнице оптимизировать расход человеческого ресурса, а пациентам – возможность более удобного взаимодействия с поликлиникой в целом.

3.3 Задачи

- описание предметной области разрабатываемого продукта;

- определение требований и критериев для оценивания надежного и качественного решения;

- выбор технологий и средств разработки для реализации поставленной цели;

- проектирование информационной системы, описание жизненного цикла пользователя и работы приложения;

- реализация системы с использованием выбранных технологий;

- тестирование системы на разных уровнях.

4 Требования к программе или программному изделию

4.1 Требования к функциональным характеристикам

- возможность авторизации и аутентификации пользователей с помощью пароля;

- возможность записи к врачу с помощью поля ввода и оставления контактных данных для обратной связи;

- изменение записи, отмена записи к врачу через личный кабинет;

4.2 Требования к надежности

- многократное использование данных;

- сохранность данных.

4.3 Требования к параметру и составу технических средств

Минимальные системные требования:

- процессор: 2 ГГц или более мощный.

- оперативная память (RAM): 1 ГБ и более.

4.5 Требования к программным средствам

Серверная часть приложения должна быть реализована с использованием языка Go. Для хранения данных использовать *PostrgreSQL*. Клиентская часть реализована с использованием фреймворка *React*. Для сборки проекта использовать *create react app*. Для контейнеризации приложений использовать Docker.

4.6 Требования к защите информации и программ

- целостность базы данных;

- идентификация и авторизация пользователя.

5 Требования к программной документации

Состав программной документации должен включать: техническое задание, пояснительную записку.