**РЕФЕРАТ**

ВКР бакалавра «Система для онлайн-консультирования с врачами. Разработка клиентской части».

Пояснительная записка: 62 стр., 32 рис., 6 табл., 28 источн., 4 прилож.

Ключевые слова: онлайн-консультации с врачами, web-приложение, JavaScript, информационная система (далее ИС), программный продукт (далее ПП).

Объектом исследования в данной работе являются системы для онлайн-консультирования с врачами.

Цель работы – разработка клиентской части web-приложения для поиска врача и проведения онлайн-консультации.

Web-приложение предназначено для пациентов желающих получить медицинскую консультацию, а так же для врачей, осуществляющих эту консультацию за оплату.

Для пациента приложение позволяет удобно выполнять поиск врача по заданным фильтрам, записываться на консультацию с выбором удобного для пациента времени, вести медицинскую карту, хранить анализы и снимки. Для врача сервис предоставляет выбирать удобный рабочий график и проводить онлайн-консультации за назначенную врачом оплату.

В системе предусмотрены мероприятия защиты персональных данных и разграничения доступа. Приложение имеет интуитивно понятный дружественный интерфейс.

Пояснительная записка состоит из введения, 4 разделов и заключения, в которых проводится анализ информационных процессов в задаче, а именно, исследование истории вопроса и его состояния на сегодняшний день, рассмотрены существующие на сегодняшний день аналоги, а также выполнена разработка и тестирование получившегося программного модуля.

**СОДЕРЖАНИЕ**

СОДЕРЖАНИЕ 5

ВВЕДЕНИЕ 6

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ, СУЩЕСТВУЮЩИХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, СИСТЕМ ИЛИ МЕТОДОВ И АЛГОРИТМОВ, КОТОРЫЕ РЕШАЮТ АНАЛОГИЧНЫЕ ЗАДАЧИ 7

1.1 Анализ предметной области и её информационные характеристики 7

1.2 Обзор существующих аналогов 7

1.3 Обоснование выбора инструментальных средств 7

1.4 Постановка задачи 7

Выводы по разделу 1 7

2 СИСТЕМОТЕХНИЧЕСКИЙ АНАЛИЗ И ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ «КЛИЕНТСКАЯ ЧАСТЬ ДЛЯ СИСТЕМЫ ОНЛАЙН-КОНСУЛЬТИРОВАНИЯ С ВРАЧАМИ» 14

2.1 Построение диаграмм потоков данных (DFD) в проектируемой системе для описания процессов документооборота и обработки информации 14

2.2 Разработка функциональных и информационных моделей IDEF0 проектируемой системы 14

2.3 Разработка структуры данных 14

2.4 Разработка математической (имитационной) модели 14

Выводы по разделу 2 14

3 РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ «КЛИЕНТСКАЯ ЧАСТЬ ДЛЯ СИСТЕМЫ ОНЛАЙН-КОНСУЛЬТИРОВАНИЯ С ВРАЧАМИ» 14

3.1 Разработка компонентов программного модуля 14

3.2 Разработка интерфейса программного модуля 14

Выводы по разделу 3 14

4 ТЕСТИРОВАНИЕ РАЗРАБОТАННОГО ПРОГРАММНОГО МОДУЛЯ «КЛИЕНТСКАЯ ЧАСТЬ ДЛЯ СИСТЕМЫ ОНЛАЙН-КОНСУЛЬТИРОВАНИЯ С ВРАЧАМИ» 20

4.1 Сценарии тестирования программного модуля 20

Выводы по разделу 4 20

ЗАКЛЮЧЕНИЕ 69

**ВВЕДЕНИЕ**

**Актуальность темы**. В связи с нынешней эпидемиологической ситуацией в мире остро стоит проблема записи на прием к врачу и получения как минимум первичной медицинской консультации, по причине того, что государственные больницы не справляются с большим количеством пациентов. Кроме того, посещение поликлиник увеличивает количество потенциально опасных контактов. Так же часто встречаются ситуации, связанные с ограничениями по месту жительства и отсутствия возможности своевременно получить консультацию специалиста.

Существуем большое количество готовых программных продуктов, однако среди них отсутствуют системы где врачи имеют возможность зарегистрироваться, пройти верификацию и предоставлять услуги консультирования пациентам.

Целью выпускной квалификационной работы является разработка web-приложения для онлайн-консультирования с врачами. В процессе достижения поставленной цели решались следующие задачи:

1) проанализировать функциональные обязанности врача, пациента и типовых вариантов реализации web-приложений;

2) разработать алгоритмы функционирования приложения, выбор платформы и языка программирования для создания web-приложения;

3) разработать структурную схему приложения, дизайн и реализовать его в виде web-приложения;

4) протестировать web-приложение.

**Объектом исследования** являются системы для онлайн-консультирования с врачами.

**Предметом исследования** настоящей работы является web-приложение для проведения онлайн-консультаций.

**Практическое значение работы.** Результаты данной работы могут представлять интерес для лечебно-профилактических учреждений общего профиля и специализированных клиник.

**Научная и практическая новизна.** Разработано оригинальное web-приложение, отличающееся широкой функциональностью, информационной безопасностью и простотой использования.

**Структура работы.** Данная работа состоит из пояснительной записки, включающей в себя введение, четыре раздела, выводы, список использованных источников и приложения.

Во введении приведено обоснование актуальности решаемой задачи и сформирована цель и задачи работы.

В первом разделе выполнен анализ информационных процессов в задаче разработки web-приложения для онлайн-консультирования с врачами. Проведен сравнительный анализ существующих web-приложений, в которых решаются схожие задачи. Выполнена постановка задачи на разработку web-приложения.

Во втором разделе разработаны алгоритмы функционирования приложения, выбрана платформа и язык программирования для создания web-приложения.

В третьем разделе разработана структурная схема приложения, пользовательский интерфейс, реализованы модули web-приложения.

В четвертом разделе выбрана методология тестирования и протестировано web-приложение. В заключении сделаны выводы во всей работе.

**1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ, СУЩЕСТВУЮЩИХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, СИСТЕМ ИЛИ МЕТОДОВ И АЛГОРИТМОВ, КОТОРЫЕ РЕШАЮТ АНАЛОГИЧНЫЕ ЗАДАЧИ**

**1.1 Анализ предметной области и её информационные характеристики**

**1.2 Обзор существующих аналогов**

**1.3 Обоснование выбора инструментальных средств**

В ходе работы на стороне клиента было решено использовать язык TypeScript и библиотеку для разработки пользовательских интерфейсов React. Для управления состоянием приложения была выбрана библиотека MobX. В качестве CSS препроцессора был взят SASS. Для организации соединения в режиме реального времени была использована библиотека Socket.IO.

TypeScript – это язык программирования, представленный Microsoft в 2012 году и позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript. TypeScript отличается от JavaScript возможностью явного статического назначения типов, поддержкой использования полноценных классов, а также поддержкой подключения модулей, что призвано повысить скорость разработки, облегчить читаемость, рефакторинг и повторное использование кода, помочь осуществлять поиск ошибок на этапе разработки и компиляции, и, возможно, ускорить выполнение программ.

React – 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. Из особенностей можно выделить следующее:

* Virtual DOM - легковесная копия DOM дерева, в которую вносятся изменения, после чего происходит сравнение DOM дерева с его виртуальной копией, определяется разница и происходит перерисовка того, что было изменено.
* JSX – расширение синтаксиса JavaScript, которое позволяет использовать HTML синтаксис для описания структуры интерфейса.
* Методы жизненного цикла, при помощи которых разработчик может описывать поведение компонента на каждом этапе его жизни (при монтировании, обновлении данных, либо его удалении).

MobX – это автономная библиотека, для управления фронтенд-состоянием приложения. MobX обеспечивает консистентность и согласованность внутреннего состояния фронтенд-приложения, предоставляя удобные инструменты для его изменения.

MobX имеет следующие преимущества по сравнению с его аналогом Redux:

* Эффективен сразу после установки, отсутствие “Многословности” по сравнению с Redux;
* Объектно-ориентированных подход;
* Упрощение работы с асинхронными действиями;
* Производительность и скорость разработки.

Sass (Syntactically Awesome Stylesheets) — это метаязык на основе CSS, предназначенный для увеличения уровня абстракции CSS-кода и упрощения файлов каскадных таблиц стилей.

Язык Sass имеет два синтаксиса:

* sass — отличается отсутствием фигурных скобок, в нём вложенные элементы реализованы с помощью отступов;
* scss (Sassy CSS) — использует фигурные скобки, как и сам CSS.

Socket.IO — это библиотека JavaScript для веб-приложений реального времени. Он обеспечивает двустороннюю связь в реальном времени между веб-клиентами и серверами. Он состоит из двух частей: клиентской библиотеки, которая запускается в браузере, и серверной библиотеки для node.js. Оба компонента имеют идентичный API.

Написание приложения для реального времени с использованием популярных стеков веб-приложений, таких как LAMP (PHP), традиционно было очень трудным. Он включает в себя опрос сервера на наличие изменений, отслеживание временных меток, и это намного медленнее, чем должно быть.

Сокеты традиционно были решением, вокруг которого строится большинство систем реального времени, обеспечивая двунаправленный канал связи между клиентом и сервером. Это означает, что сервер может отправлять сообщения клиентам. Всякий раз, когда происходит событие, идея заключается в том, что сервер получит его и отправит заинтересованным подключенным клиентам.

Socket.IO довольно популярен, его используют Microsoft Office, Yammer, Zendesk, Trello и многие другие организации для создания надежных систем реального времени. Это одна из самых мощных JavaScript-фреймворков на GitHub и наиболее зависимая от модуля NPM (Node Package Manager). Socket.IO также имеет огромное сообщество, что означает, что найти помощь довольно легко.

**2 СИСТЕМОТЕХНИЧЕСКИЙ АНАЛИЗ И ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ «КЛИЕНТСКАЯ ЧАСТЬ ДЛЯ СИСТЕМЫ ОНЛАЙН-КОНСУЛЬТИРОВАНИЯ С ВРАЧАМИ»**

**2.1 Построение диаграмм потоков данных (DFD) в проектируемой системе для описания процессов документооборота и обработки информации**

В системе онлайн консультаций с врачами основным процессом является регистрация врачей/пациентов, поиск врачей, запись на консультацию и ее последующее проведение. Для своей работы система использует внешние сущности: врач и пациент.

Пациент определяет свою проблему сам находит врача с помощью фильтров или поиска, далее оплачивает услугу и записывается на консультацию. После прохождения консультации пациент получает результат в виде рекомендации по лечению.

При регистрации врача происходит проверка подлинности документов о квалификации врача модератором, если ответ положительный, то врач может перейти к процессу консультаций пациентов, которые записались к нему на прием.

Основной процесс включает в себя следующие процессы: регистрация, запись на консультацию, поиск врача пациентом, консультация, составление отзыва после консультации, перевод денег врачу.

На рисунках 2.1-2.4 представлены DFD-диаграммы полученные в ходе проектирования системы.

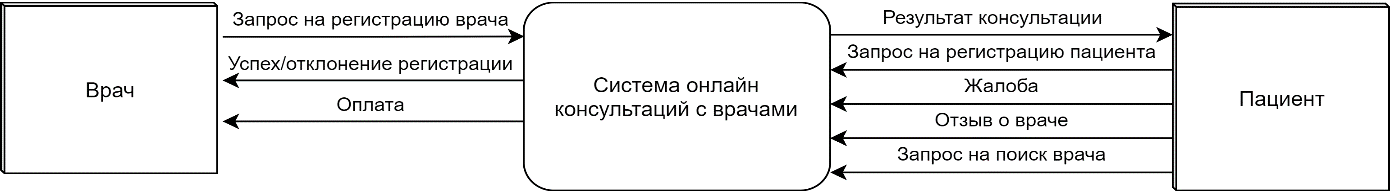
****

Рисунок 2.1 – DFD-диаграмма основного процесса

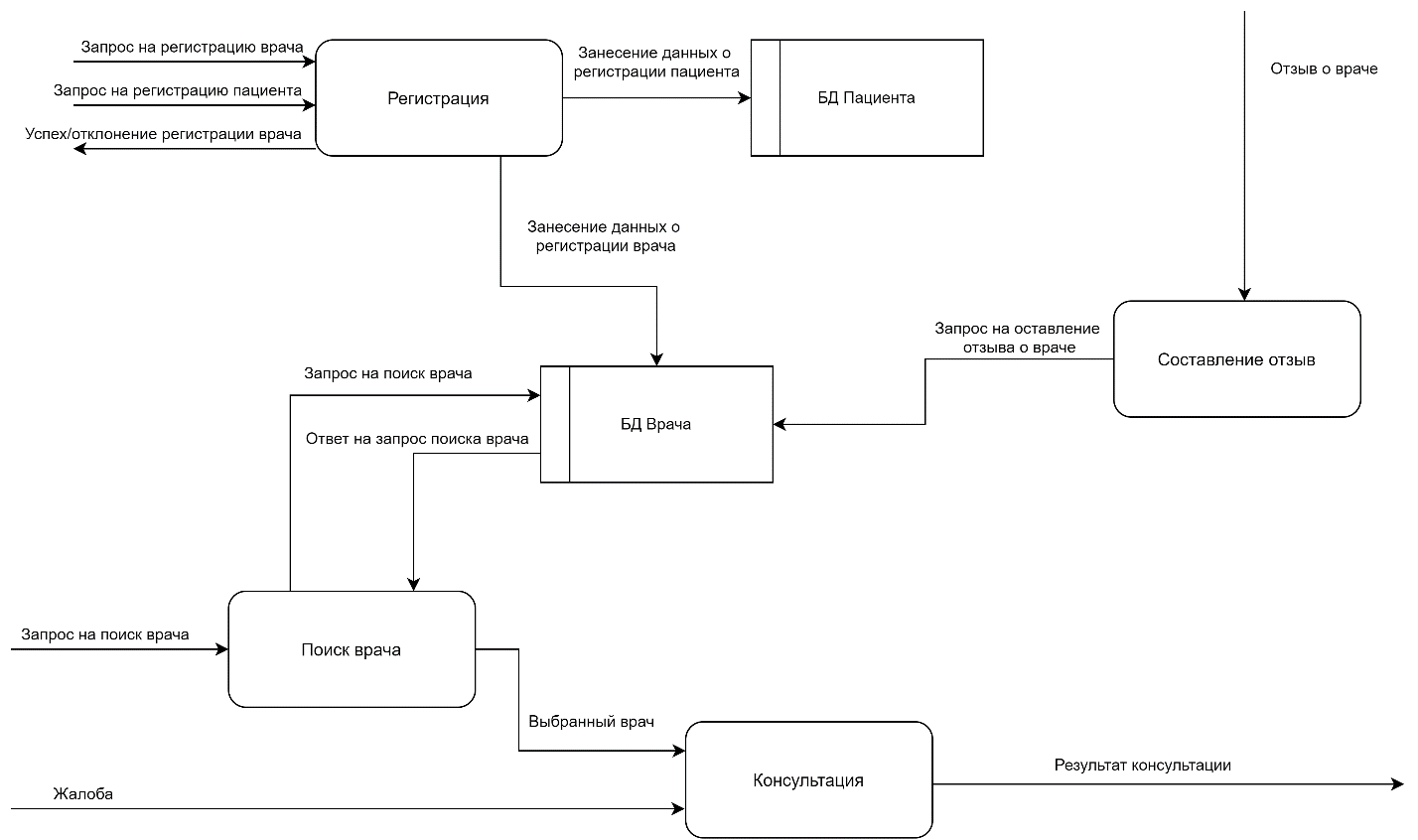
****

Рисунок 2.2 – DFD-диаграмма декомпозиции основного процесса

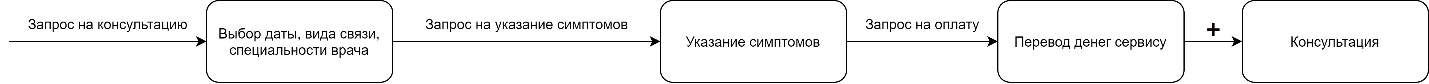
****

Рисунок 2.3 – DFD-диаграмма декомпозиции процесса консультации

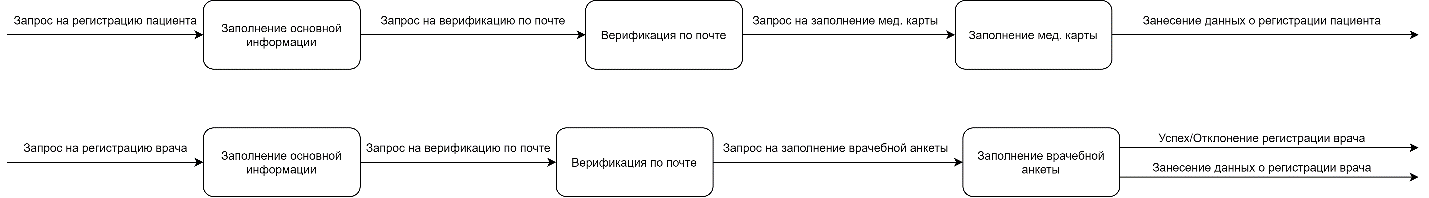
****

Рисунок 2.4 – DFD-диаграмма декомпозиции процесса регистрации

**2.2 Разработка функциональных и информационных моделей IDEF0-IDEF1 проектируемой системы**

Таблица 2.1 – Процессы диаграммы A1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Шифр | Название процесса | Входные данные | Управляющие данные | Механизм | Результат процесса |
| A1 | Предоставить услуги консультации | Запрос на регистрацию врача и пациента, | Прайс лист, установленное время, форма регистрации и тестирования | Врач, пациент, модератор | Результат консультации, оплата, отклонение оплаты, отклонение регистрации врача |

Таблица 2.2 – Декомпозиция процесса A1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Шифр | Название процесса | Входные данные | Управляющие данные | Механизм | Результат процесса |
| A11 | Регистрация врача | Запрос на регистрацию врача | Форма регистрации | Врач, модератор | Отклонение регистрации врача |
| A12 | Регистрация пациента | Запрос на регистрацию пациента | Форма регистрации | Пациент, модератор | Жалоба |
| A13 | Поиск врача | Жалоба | Форма тестирования | Пациент, врач | Выбранный врач |
| A14 | Оплачивать | Выбранный врач | Прайс лист | Пациент | Чек об оплате |
| A15 | Консультация | Чек об оплате | Установленное время | Врач, пациент | Результат консультации |
| A16 | Составить отзыв | Результат консультации | Результат консультации | Пациент | Отклонение оплаты |
| A17 | Перевод врачу | Положительный отзыв | Положительны отзыв | Модератор | Оплата |

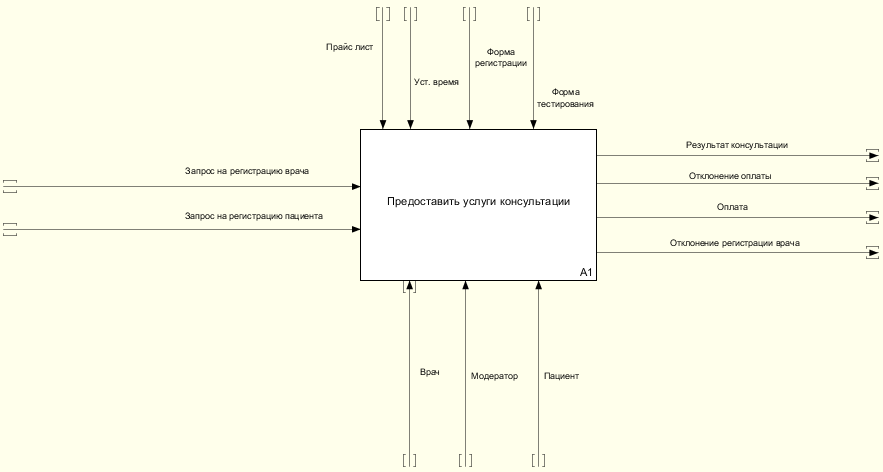


Рисунок 2.5 – IDEF0-диаграмма основного процесса



Рисунок 2.6 – IDEF0-диаграмма декомпозиции основного процесса

**2.3 Разработка структуры данных**

Написать про сторы

**3 РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ «КЛИЕНТСКАЯ ЧАСТЬ ДЛЯ СИСТЕМЫ ОНЛАЙН-КОНСУЛЬТИРОВАНИЯ С ВРАЧАМИ»**

**3.1 Разработка компонентов программного модуля**

При разработки клиентской части использовалась библиотека для разработки пользовательских интерфейсов React. На рисунке 18 представлена структура клиентской части.

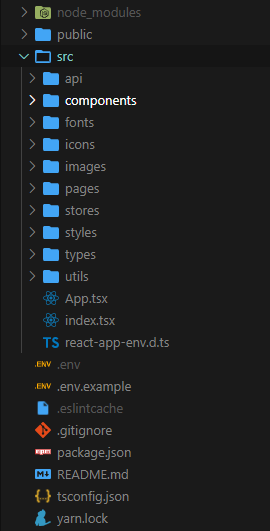


Рисунок 18 – Структура клиентского приложения

В папка images и icons содержатся картинки и иконки всего приложения.

Так как в работе использовался препроцессор для стилей SCSS, то в папке styles содержатся его файлы, разделенные на файл с переменными и файл с глобальными стилями, а также ts-файл с описанием конфигурации Material UI.

Папки utils и api содержат вспомогательные функции, участвующие в дальнейшей разработке, такие как описание api методов, функции для работы со строками и массивами, константы и файл с базовыми настройками библиотеки axios.

В папке fonts соответственно содержатся шрифты, используемые во всем приложении.

Так как в работе использовалась библиотека MobX для управления состоянием приложения, то в проекте находится папка stores, которая содержит подпапку interfaces для определения интерфейсов хранилищ и сами классы сторов, объекты которых создаются в приведенном ниже фрагменте кода из файла index.ts:

*export* class RootStore implements IStores {

    routerStore: IRouterStore;

    signUpStore: ISignUpStore;

    signInStore: ISignInStore;

    userStore: IUserStore;

    modalsStore: IModalsStore;

    questionnaireStore: IQuestionnaireStore;

    specialtiesStore: ISpecialtiesStore;

    doctorStore: IDoctorStore;

    homeStore: IHomeStore;

    constructor(history: History) {

*this*.routerStore = new RouterStore(history);

*this*.signUpStore = new SignUpStore(*this*);

*this*.signInStore = new SignInStore(*this*);

*this*.userStore = new UserStore();

*this*.modalsStore = new ModalsStore();

*this*.questionnaireStore = new QuestionnaireStore(*this*);

*this*.specialtiesStore = new SpecialtiesStore();

*this*.doctorStore = new DoctorStore(*this*);

*this*.homeStore = new HomeStore();

    }

}

Также в проекте имеется папка components для хранения общих компонентов всего приложения, таких как: Header, Footer, Loader, Button, Dialog и т.п.

Все страницы приложения также хранятся в папке pages, структура которых приведена на рисунке 19.



Рисунок 19 – Структура папки pages

Как видно из скриншота выше, папка pages состоит из соответствующих подпапок для каждой страницы, а также в каждой из них есть папка components с локальными компонентами, которые используются только на этой странице.

Старт приложения начинается с файла index.ts, который находится в корневой папке src. В нем компонент App оборачивается в провайдеры и все приложение рендерится у блока с классом root. Фрагмент данного файла представлен ниже:

*import* React *from* "react";

*import* ReactDOM *from* "react-dom";

*import* { Router } *from* "react-router-dom";

*import* { createBrowserHistory, History } *from* "history";

*import* { MuiThemeProvider } *from* "@material-ui/core";

*import* { MuiPickersUtilsProvider } *from* "@material-ui/pickers";

*import* DateFnsUtils *from* "@date-io/date-fns";

*import* ruLocale *from* "date-fns/locale/ru";

*import* { App } *from* "./App";

*import* { RootStore } *from* "./stores";

*import* { StoreProvider } *from* "./stores/useStore";

*import* { main } *from* "./styles/material";

*import* "./styles/index.scss";

const history: History = createBrowserHistory();

const rootStore = new RootStore(history);

ReactDOM.render(

    <StoreProvider *store*={rootStore}>

        <MuiThemeProvider *theme*={main}>

            <MuiPickersUtilsProvider *utils*={DateFnsUtils} *locale*={ruLocale}>

                <Router *history*={history}>

                    <App />

                </Router>

            </MuiPickersUtilsProvider>

        </MuiThemeProvider>

    </StoreProvider>,

    document.getElementById("root")

);

Сам компонент App содержит jsx-разметку глобальных компонентов, конфигурация роутинга и проверку на авторизованного пользователя, в случае авторизации которого при каждом запуске приложения будет отрисовываться лоадер, тем самым скрывая видимость выполнения запроса на получения пользователя. Фрагмент файла App.tsx представлен ниже:

*export* const App: React.FC = observer(() => {

    const { userStore } = useStores();

    const { isAuthorized, pending, fetchUser } = userStore;

    useEffect(() => {

*if* (localStorage.getItem("accessToken")) {

            fetchUser();

        }

    }, [fetchUser]);

*if* (pending) {

*return* <Backdrop />;

    }

*return* (

        <React.Fragment>

            <CssBaseline />

            <ScrollHandler />

            <Switch>

                <Route *exact* *path*="/ui-kit" *component*={UiKitPage} />

                <Route

*exact*

*path*={["/", "/home", "/sign-up-confirmation"]}

*component*={HomePage}

                />

                <Route

*exact*

*path*={["/doctors", "/doctors/:page"]}

*component*={DoctorsPage}

                />

                <Route *exact* *path*="/doctor/:id" *component*={DoctorPage} />

                <PrivateRoute

*exact*

*path*="/sign-up"

*component*={SignUpPage}

*isAuthorized*={!isAuthorized}

                />

                <PrivateRoute

*exact*

*path*="/appointment"

*component*={AppointmentPage}

*isAuthorized*={isAuthorized}

                />

                <PrivateRoute

*exact*

*path*="/questionnaire"

*component*={QuestionnairePage}

*isAuthorized*={isAuthorized}

                />

                <PrivateRoute

*path*="/dashboard"

*component*={DashboardPage}

*isAuthorized*={isAuthorized}

                />

                <Route *component*={ErrorPage} />

            </Switch>

            <DialogSignIn />

            <DialogReset />

            <DialogEmail />

        </React.Fragment>

    );

});

Основные классы для взаимодействия с серверной части находятся в папке api, классы DoctorApi, QuestionnaireApi, SignIn, SignUp и UserApi представлены ниже:

*export* class DoctorApi {

    static getDoctor(id: number) {

*return* axiosInstance.get(`/api/v1/doctor/info?id=${id}`);

    }

    static getDoctors(page: number, count: number = 3) {

*return* axiosInstance.get(

            `/api/v1/doctor/paginate?page=${page}&count=${count}`

        );

    }

    static getDoctorsByCount(count: number = 7) {

*return* axiosInstance.get(`/api/v1/doctor/most-experienced?count=${count}`);

    }

}

*export* class QuestionnaireApi {

    static sendPatient(postData: IPatientQuestionnairePostData) {

*return* axiosInstance.post("/api/v1/patient/profile/questionnaire", postData);

    }

    static sendDoctor(postData: FormData) {

*return* axiosInstance.post("/api/v1/doctor/profile/questionnaire", postData);

    }

}

*export* class SignInApi {

    static signIn(postData: ISignInPostData) {

*return* axiosInstance.post("/api/v1/auth/sign-in", postData);

    }

}

*export* class SignUpApi {

    static signUp(postData: ISignUpPostData) {

*return* axiosInstance.post("/api/v1/auth/sign-up", postData);

    }

    static sendMail(postData: ISendMailPostData) {

*return* axiosInstance.post("/api/v1/auth/send-email-with-token", postData);

    }

}

*export* class UserApi {

    static getUser() {

*return* axiosInstance.get("/api/v1/user/");

    }

}

Данные методы вызываются в методах store-классов, один из них представлен ниже:

const INITIAL\_SIGN\_IN\_FORM: ISignInForm = {

    email: "",

    password: ""

};

*export* class SignInStore implements ISignInStore {

    signInForm = INITIAL\_SIGN\_IN\_FORM;

    submissionError: string | undefined = undefined;

    pending: boolean = false;

    private rootStore: IStores;

    constructor(rootStore: IStores) {

*this*.rootStore = rootStore;

        makeObservable(*this*, {

            signInForm: observable,

            submissionError: observable,

            pending: observable,

            doSignIn: action,

            setFormValue: action,

            resetForm: action

        });

    }

    doSignIn = () => {

*this*.pending = true;

*this*.submissionError = undefined;

        const postData: ISignInPostData = {

            email: *this*.signInForm.email,

            password: *this*.signInForm.password

        };

        SignInApi.signIn(postData)

            .then(

                action(({ data }: AxiosResponse<ISignInSuccessResponse>) => {

                    localStorage.setItem("accessToken", data.data.accessToken);

*this*.rootStore.userStore.fetchUser();

*this*.resetForm();

*this*.rootStore.modalsStore.setModalIsOpen("sign-in", false);

                })

            )

            .catch(

                action((error: AxiosError<ISignInErrorResponse>) => {

*this*.submissionError = error.response?.data.message;

                })

            )

            .finally(

                action(() => {

*this*.pending = false;

                })

            );

    };

    setFormValue = <K extends KeysOfSignInForm>(key: K, value: ISignInForm[K]) => {

*this*.signInForm[key] = value;

    };

    resetForm = () => {

*this*.signInForm = INITIAL\_SIGN\_IN\_FORM;

*this*.submissionError = undefined;

    };

}

А также интерфейс этого класса, находящийся в подпапке interfaces:

*export* interface ISignInStore {

    signInForm: ISignInForm;

    submissionError: string | undefined;

    pending: boolean;

    doSignIn: () => void;

    setFormValue: <K extends KeysOfSignInForm>(

        key: K,

        value: ISignInForm[K]

    ) => void;

    resetForm: () => void;

}

*export* interface ISignInForm {

    email: string;

    password: string;

}

*export* type KeysOfSignInForm = keyof ISignInForm;

Таким образом была описана основная структура клиентского приложения и приведены фрагменты кода основных компонентов и классов приложения.

**3.2 Разработка интерфейса программного модуля**

При запуске web-приложения открывается главная страница, на которой представлена основная информация о нашем сервисе (Рисунок 3) имеется секция с наилучшими специалистами (Рисунок 2) и раздел с отзывами пациентов о сервисе (Рисунок 4).

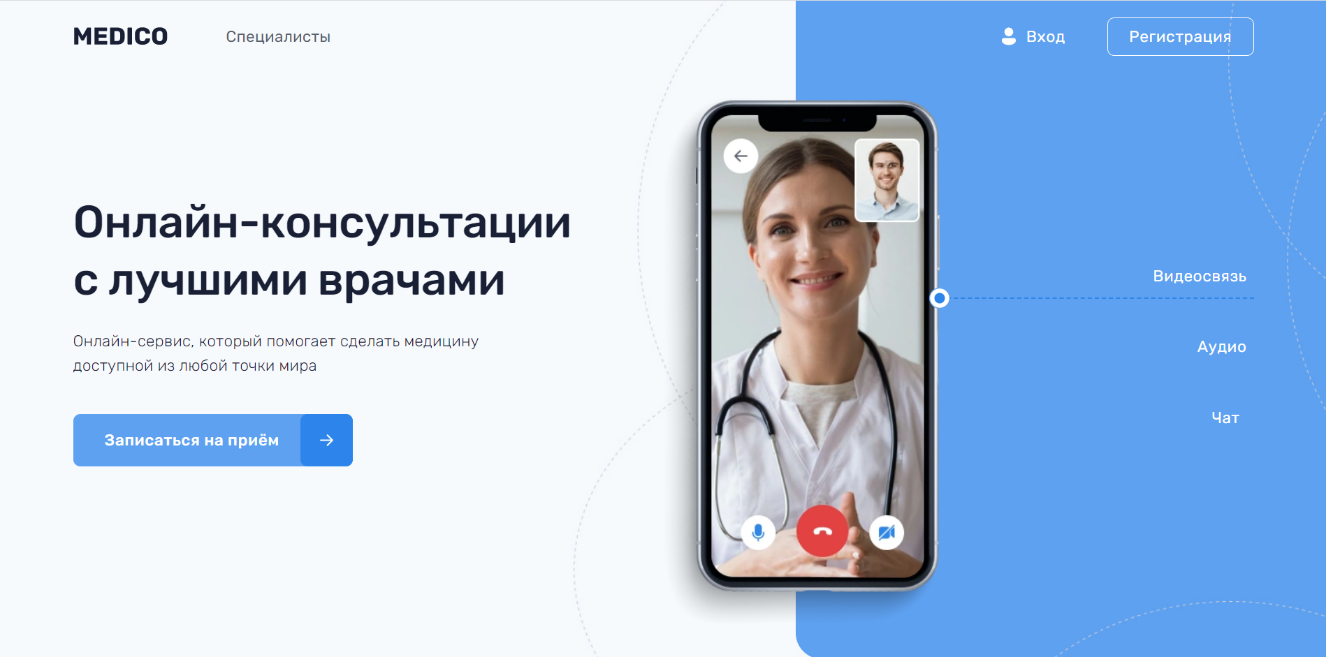


Рисунок 1 – Главный экран

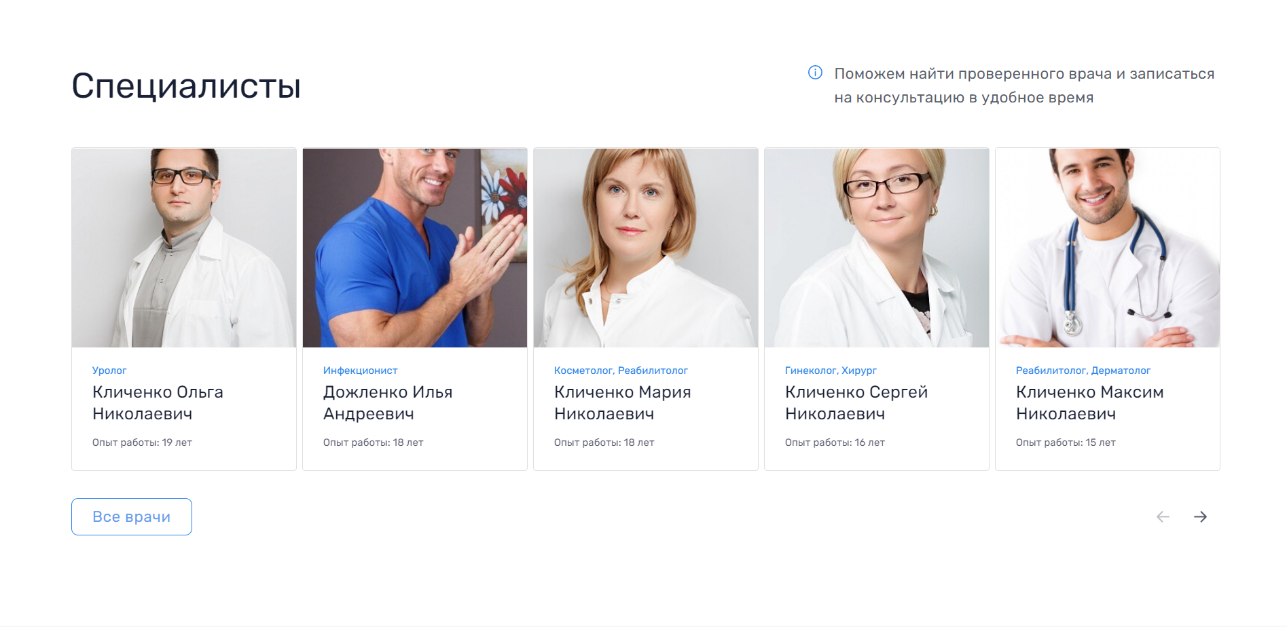


Рисунок 2 – Секция со специалистами



Рисунок 3 – Информация о сервисе

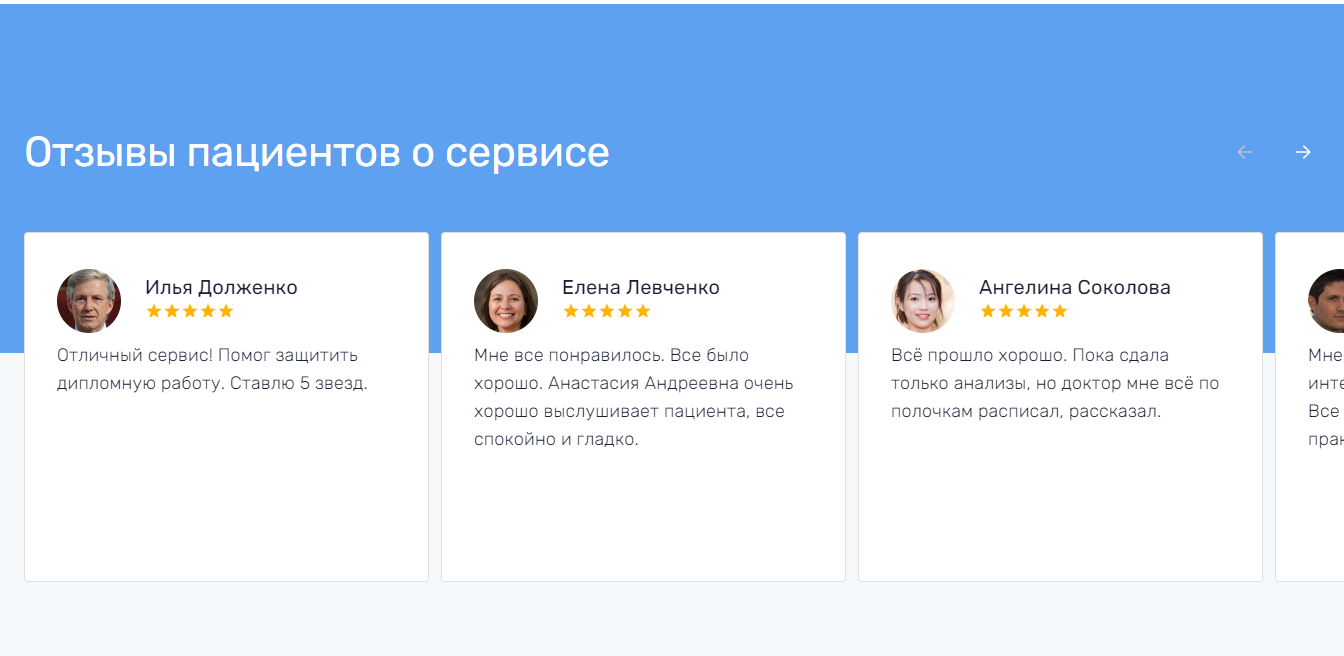


Рисунок 4 – Раздел с отзывами о сервисе

Снизу главной страницы располагается форма обратной связи, представленная на рисунке 5.

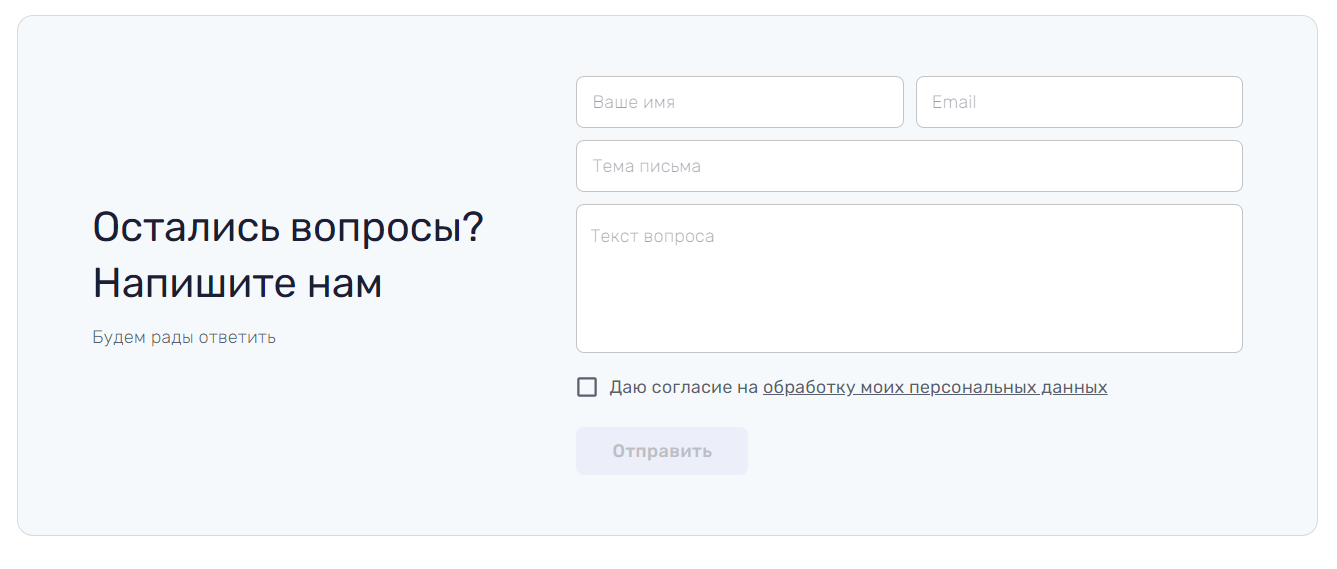


Рисунок 5 – Форма обратной связи

При нажатии на кнопку «Войти» открывается модальное окно с формой входа состоящие из полей «Email» и «Пароль» (Рисунок 5). Так же на этом окне находится кнопка восстановления пароля и регистрации (Рисунок 6)

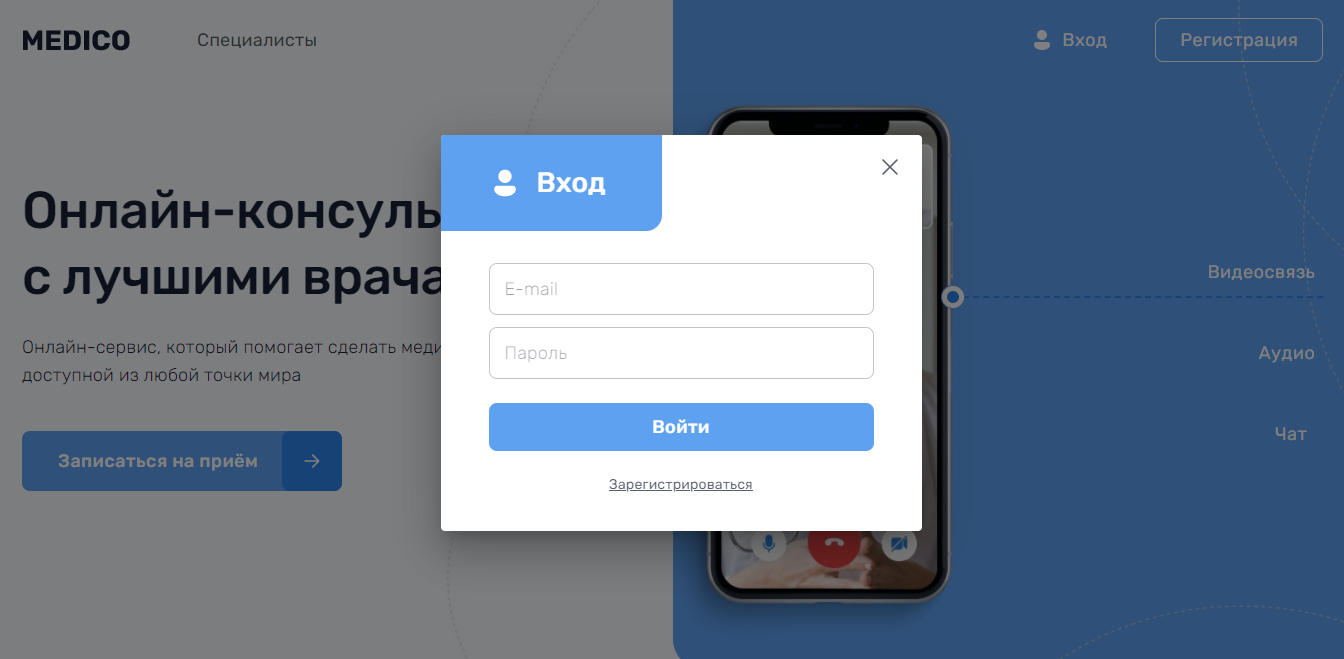


Рисунок 5 – Модальное окно входа

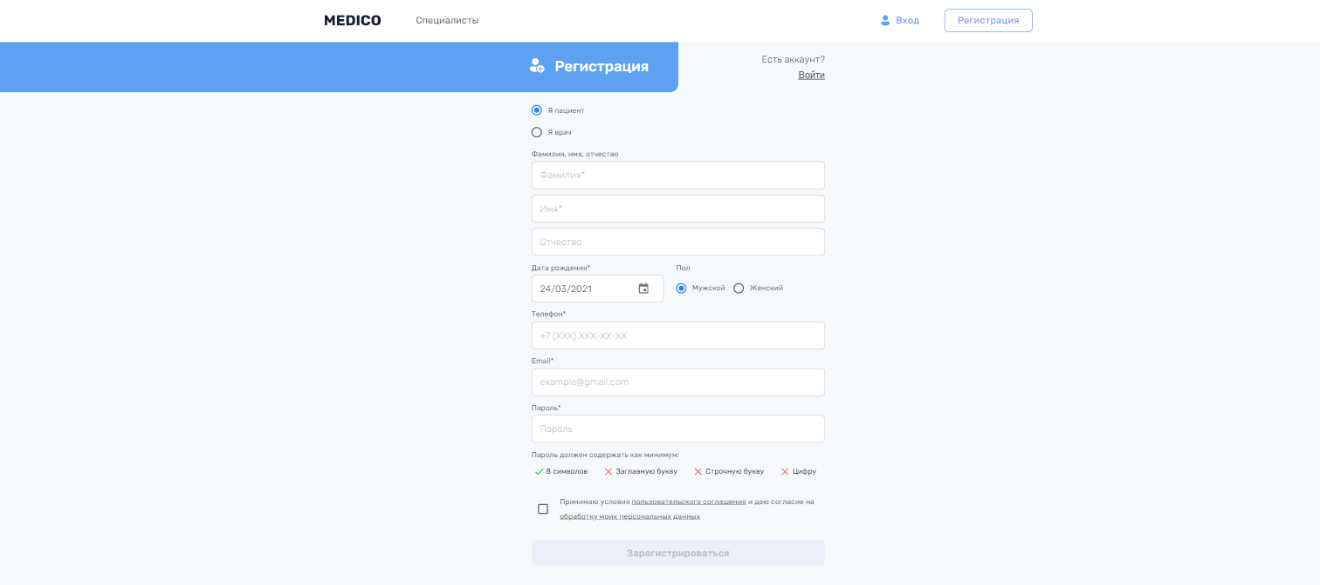


Рисунок 6 – Страница регистрации

После регистрации на почту пользователю приходит письмо с подтверждением аккаунта (Рисунок 7), и появляется модальное окно сообщающее об этом (Рисунок 8).

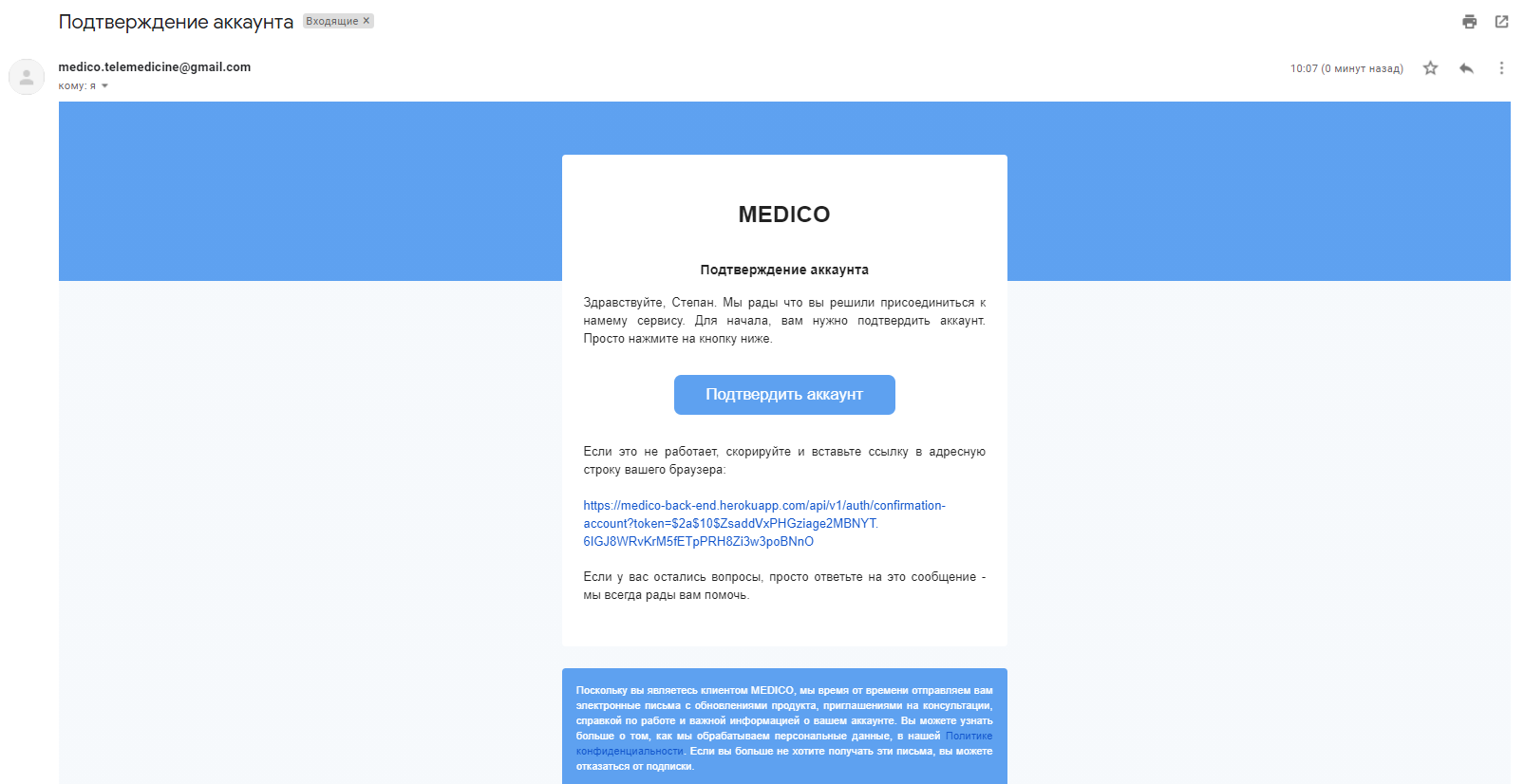


Рисунок 7 – Письмо на почте

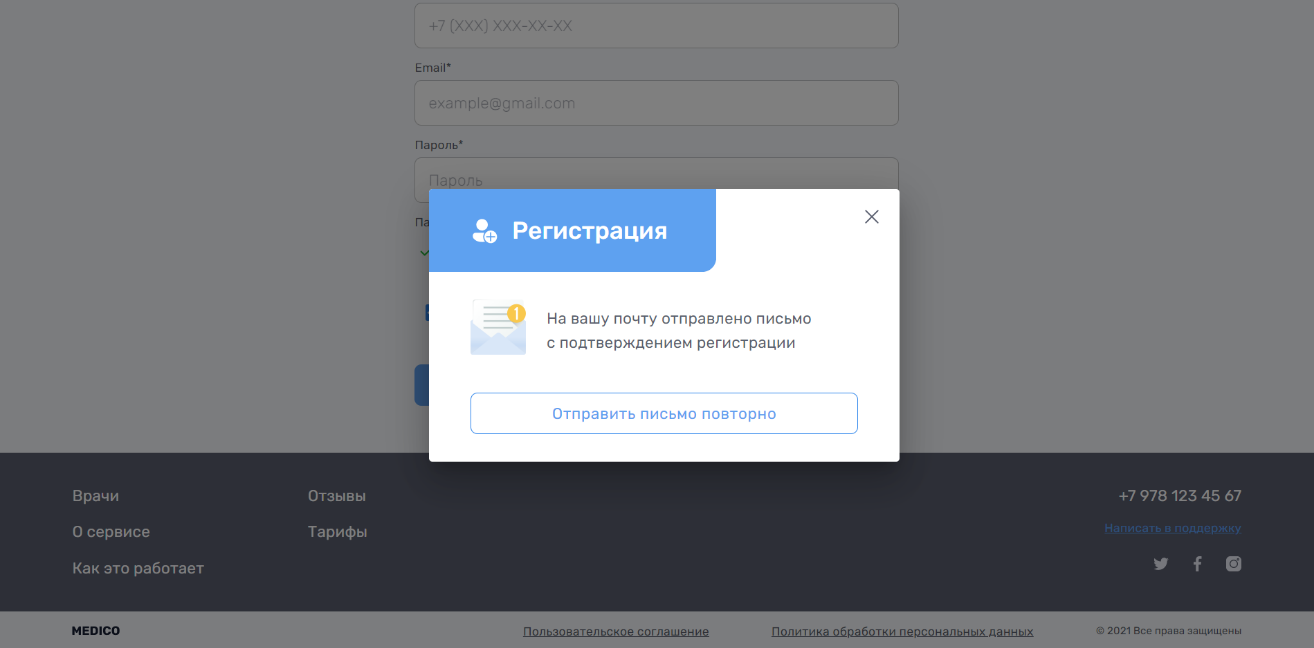


Рисунок 8 – Сообщение об отправки письма

После нажатия на кнопку «Подтвердить аккаунт» в письме пользователь перенаправляется на web-приложение (Рисунок 9).

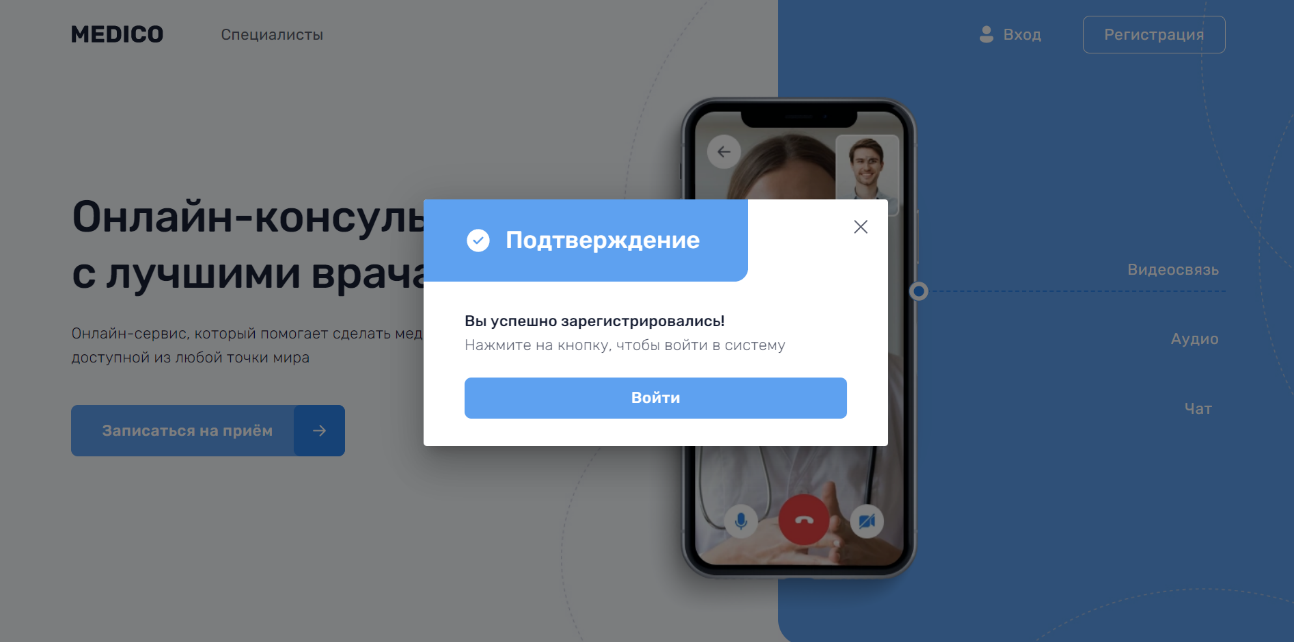


Рисунок 9 – Сообщение об успешном подтверждении аккаунта

После авторизации пациенту нужно заполнить анкету чтобы иметь доступ к личному кабинету (Рисунки 10-11).

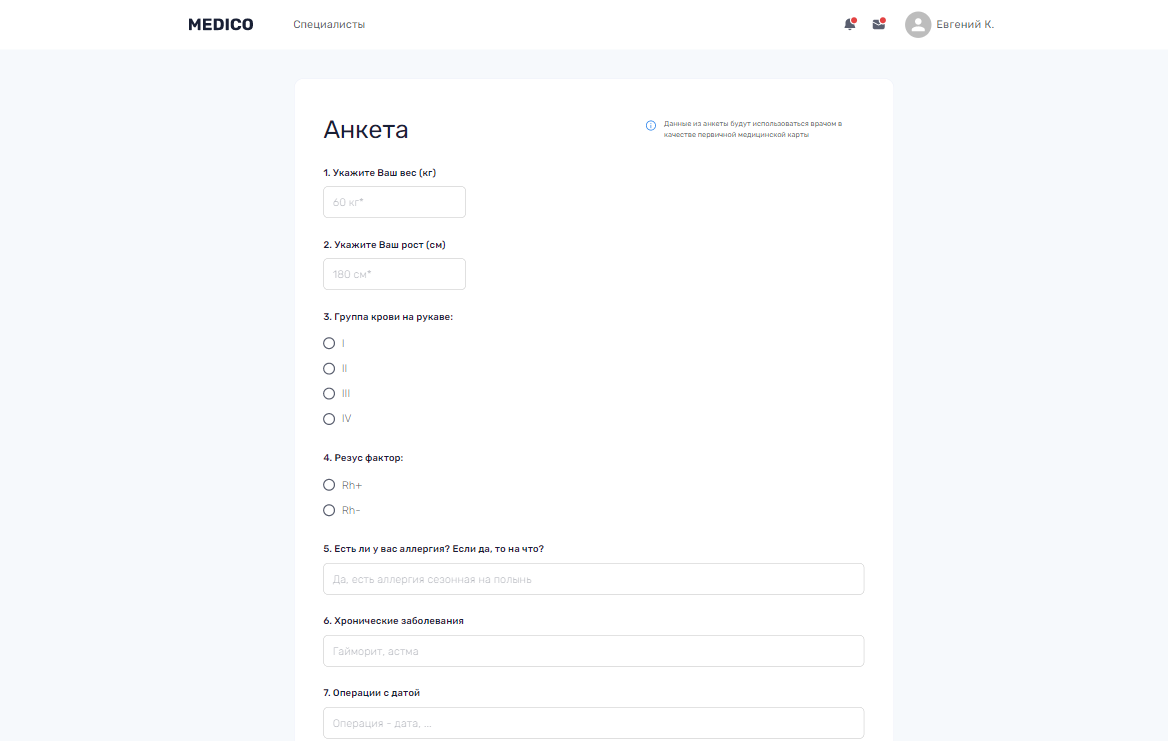


Рисунок 10 – Анкета пациента

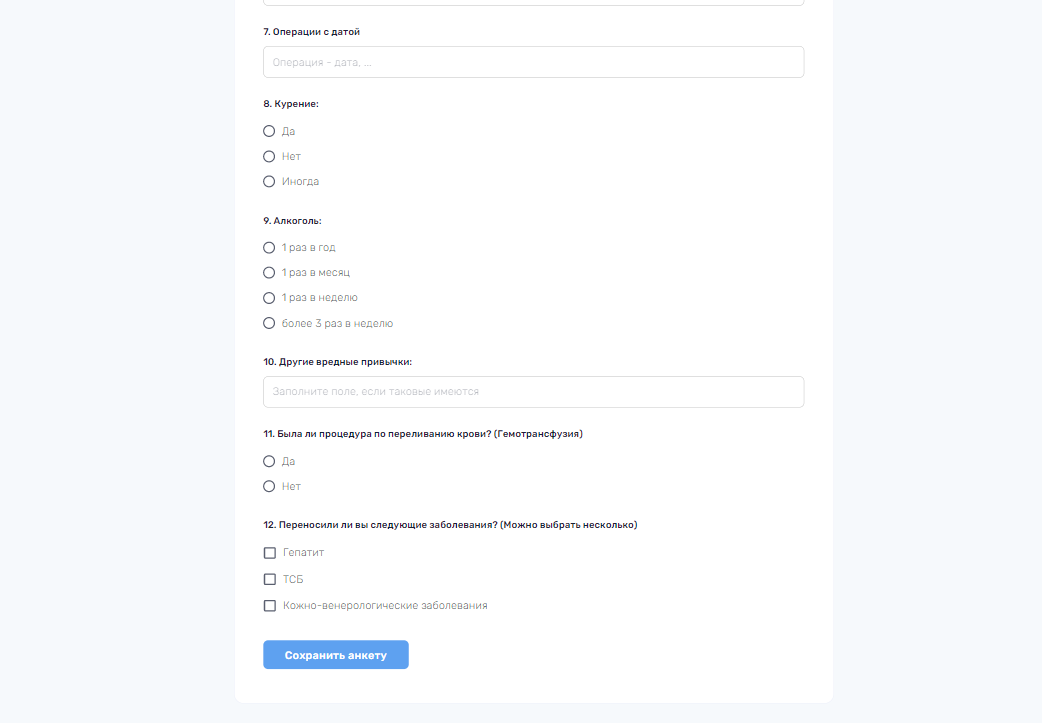


Рисунок 11 – Анкета пациента

Если пользователь является доктором, то ему необходимо заполнить заявку на врача чтобы модератор мог подтвердить его (Рисунки 12-13).

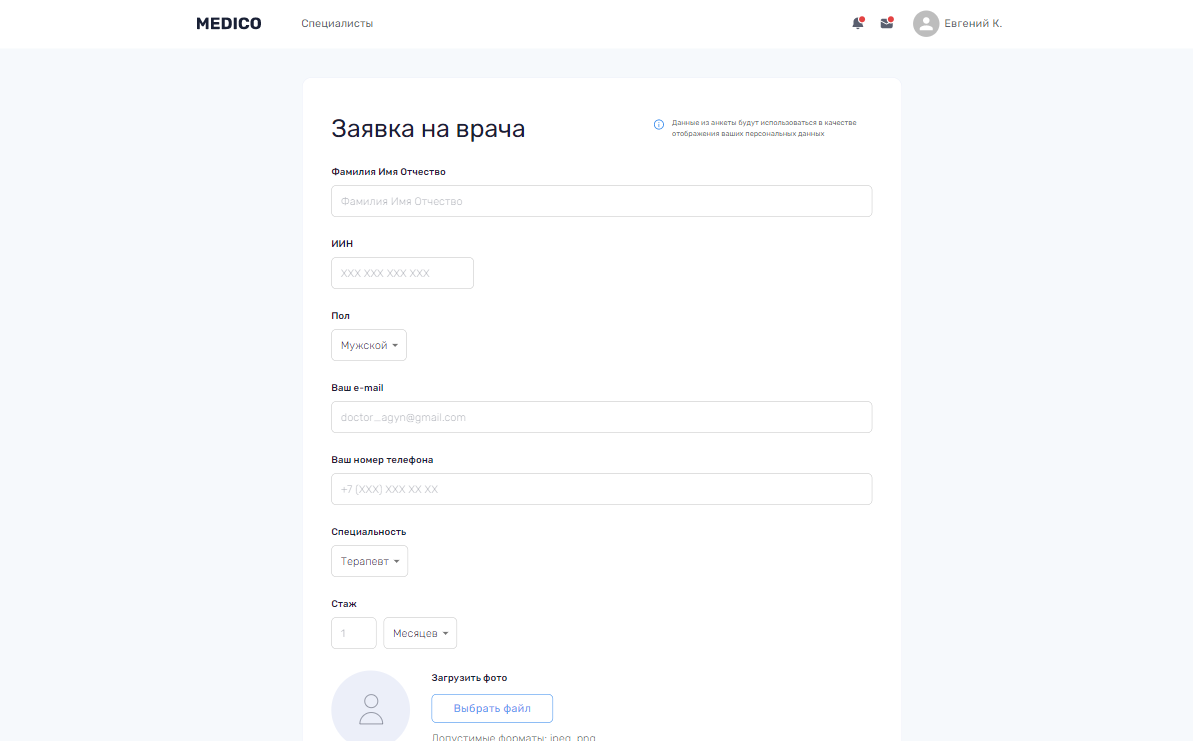


Рисунок 12 – Заявка на врача

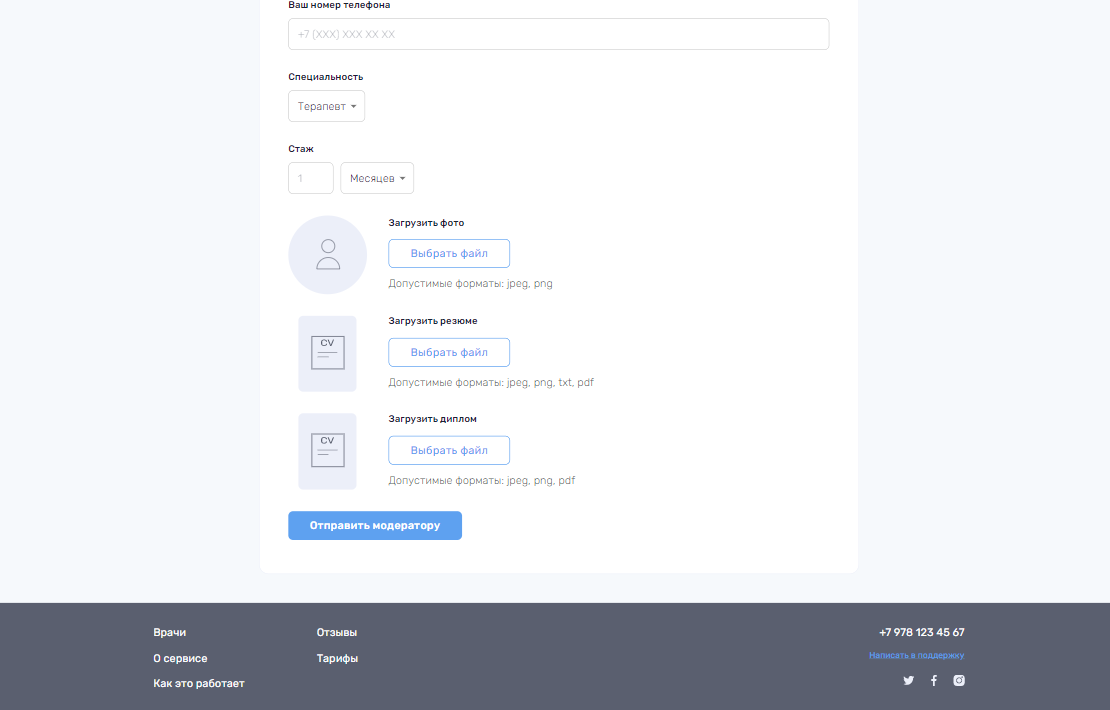


Рисунок 13 – Заявка на врача

После заполнения анкеты пациент имеет возможность найти врача с дальнейшей записью на прием (Рисунки 14-15).

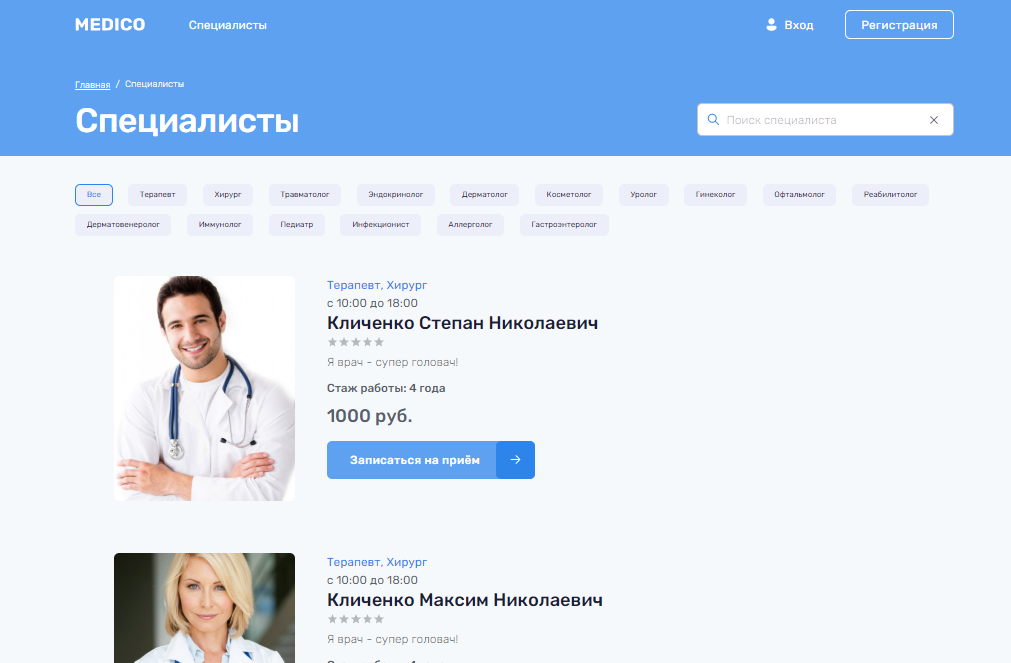


Рисунок 14 – Страница со специалистами (часть 1)

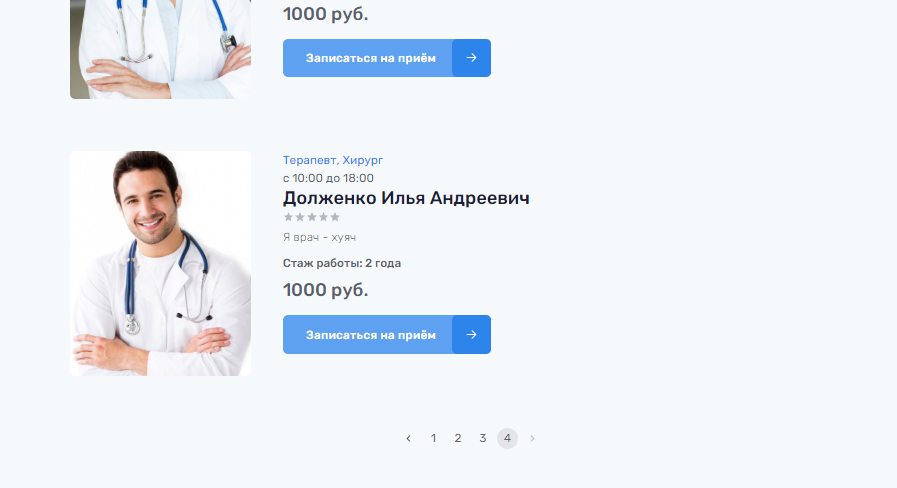


Рисунок 15 – Страница со специалистами (часть 2)

На рисунках 16-17 представлена детальная страница врача, на которой можно посмотреть отзывы пациентов, образование, опыт работы и основные направления по которым работает врач.

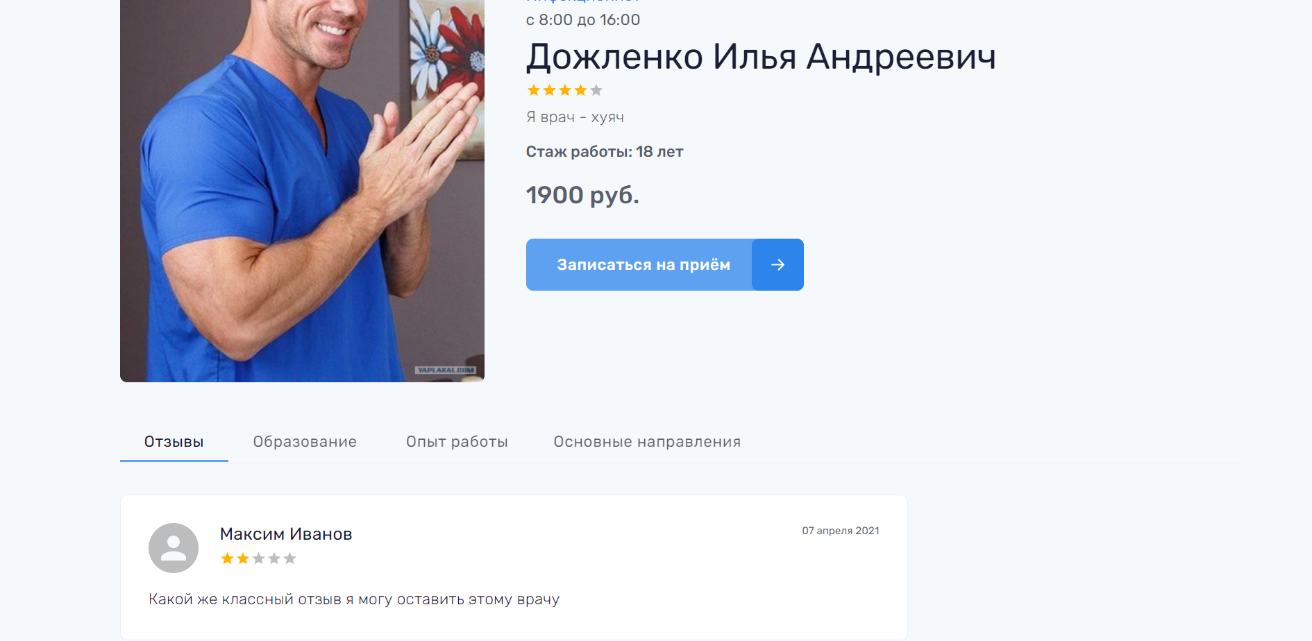


Рисунок 16 – Детальная страница врача

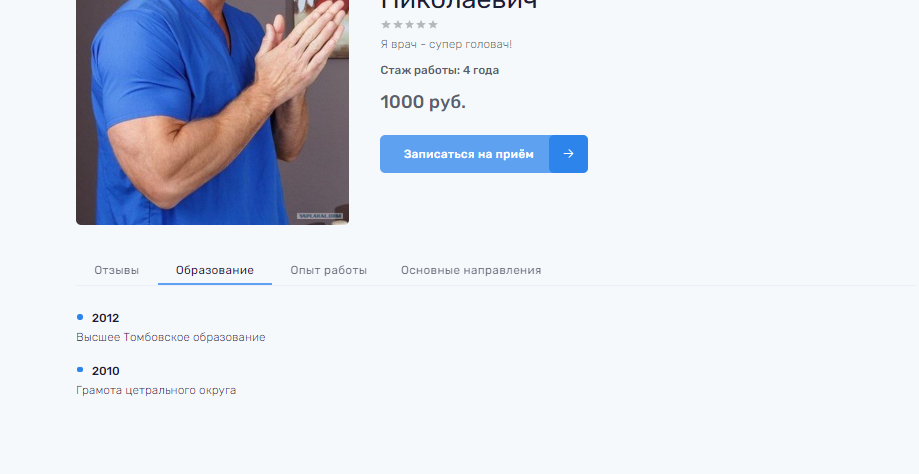


Рисунок 17 – Детальная страница врача

Если пользователь авторизован и является пациентом, то кнопка “Записаться на приём” активна и при клике на нее открывается первый этап записи на консультацию с выбором даты, времени, способом связи и необходимой специальностью. Весь процесс записи состоит из четырех этапов. Первый из них представлен на рисунке 18.

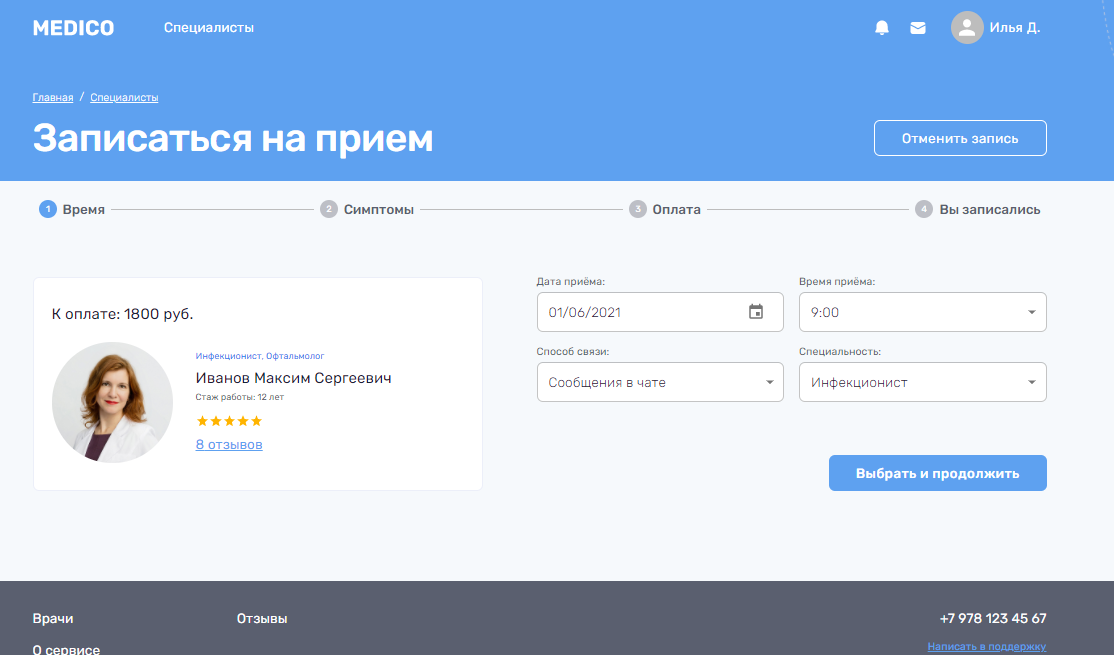


Рисунок 18 – Первый этап записи на консультацию

На втором этапе, представленном на рисунке 19, пациенту представляется возможность заполнить свои симптомы.

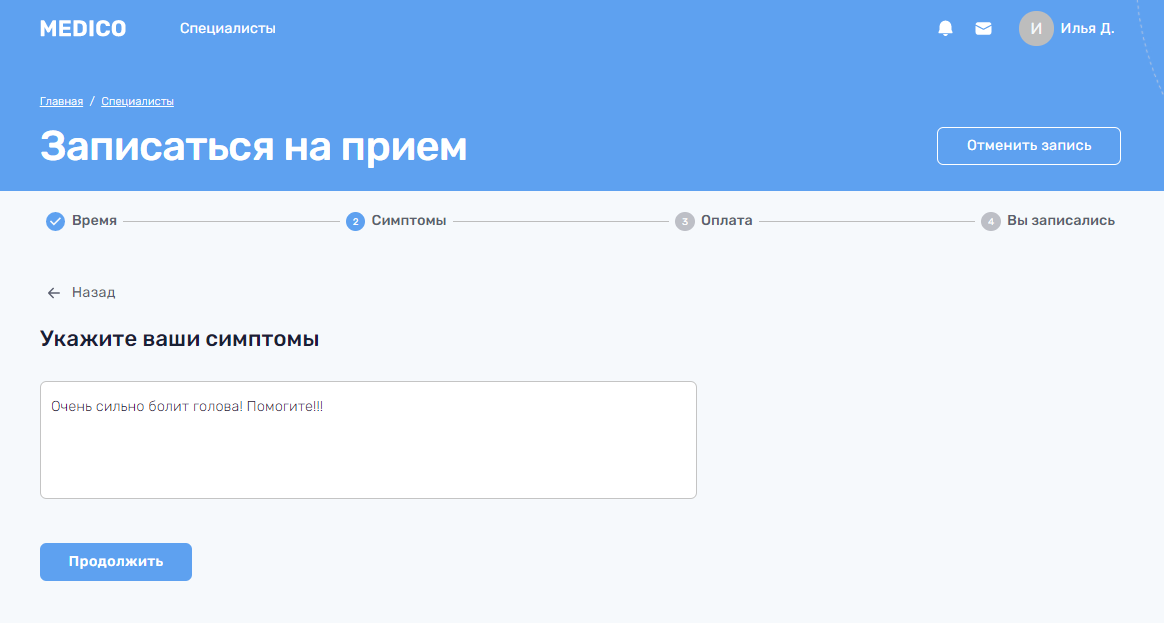


Рисунок 19 – Второй этап записи на консультацию

На третьем этапе пациенту предоставляется возможность удостовериться в введенных данных и оплатить консультацию. Данный этап представлен на рисунке 20.

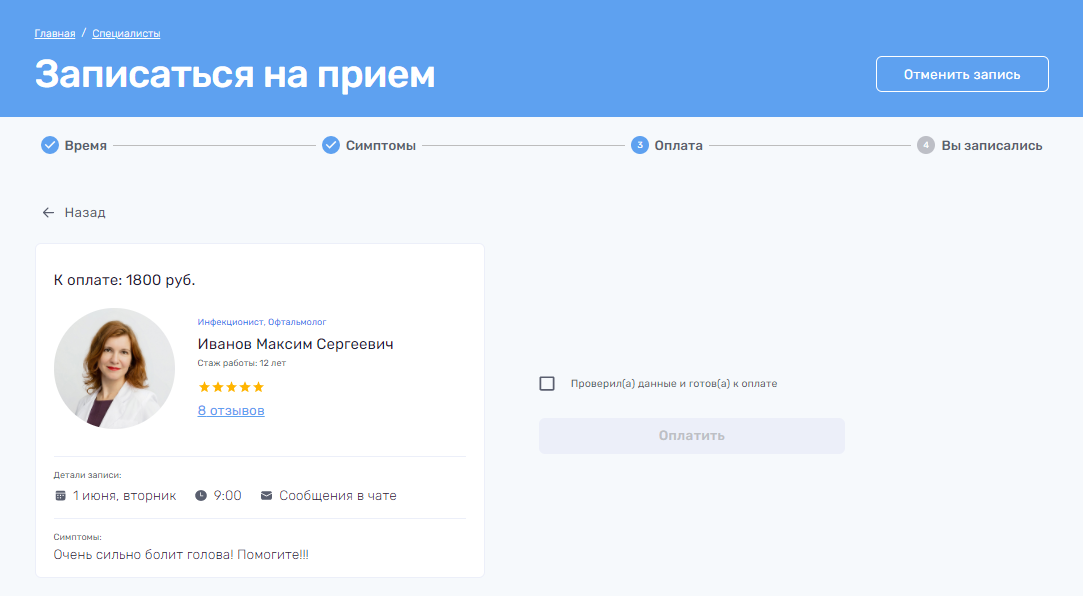


Рисунок 20 – Третий этап записи на консультацию

На завершающем этапе изображен результат записи на консультацию, также продублирована основная информация записи: имя специалиста, дата, время и способ коммуникации. Завершающий этап изображен на рисунке 21.

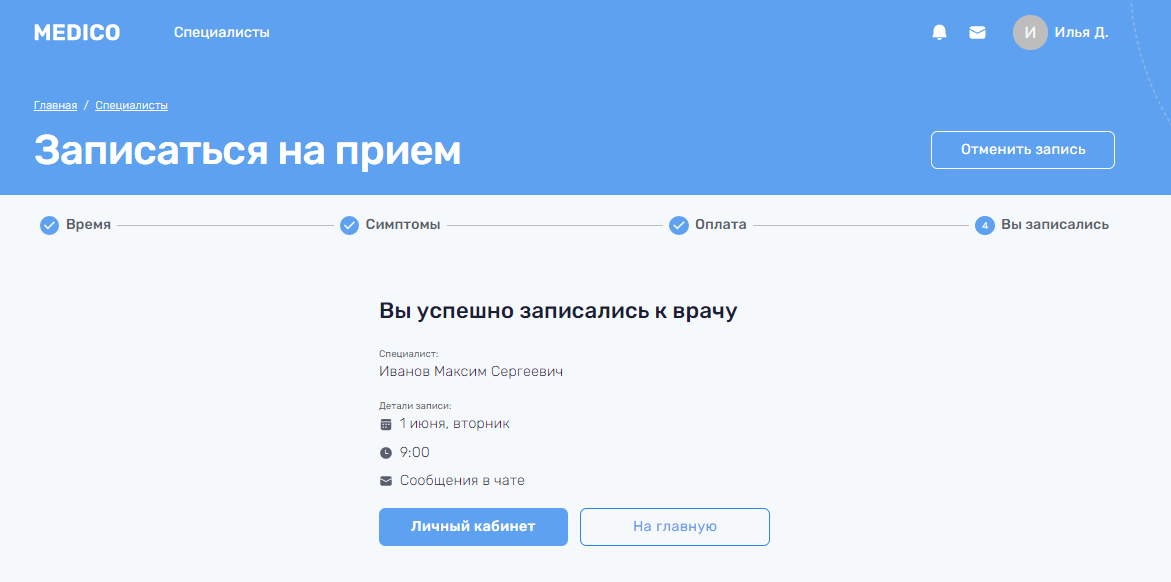


Рисунок 21 – Завершающий этап записи на консультацию

Дальнейший юзерфлоу пациента представляет из себя работу в личном кабинете, который состоит из следующих страниц: мои записи, назначения, мои анализы, мед. карта, сообщения и настройки.

Страница “Мои записи” (рисунок 22) представляет из себя три списка: активные консультации, запланированные и история.

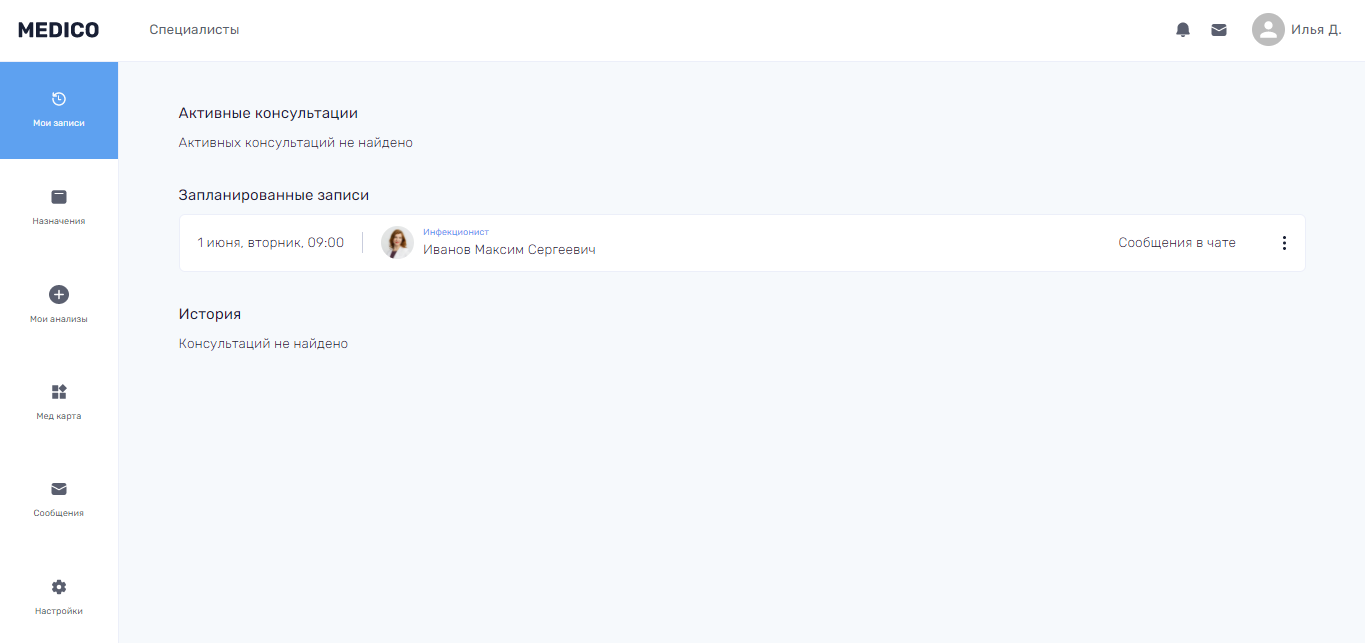


Рисунок 22 – Страница записей у пациента

Следующая страница “Назначения” содержит список рекомендаций от врача, заполняемых после процесса консультации с пациентом. Данная страница представлена на рисунке 23.

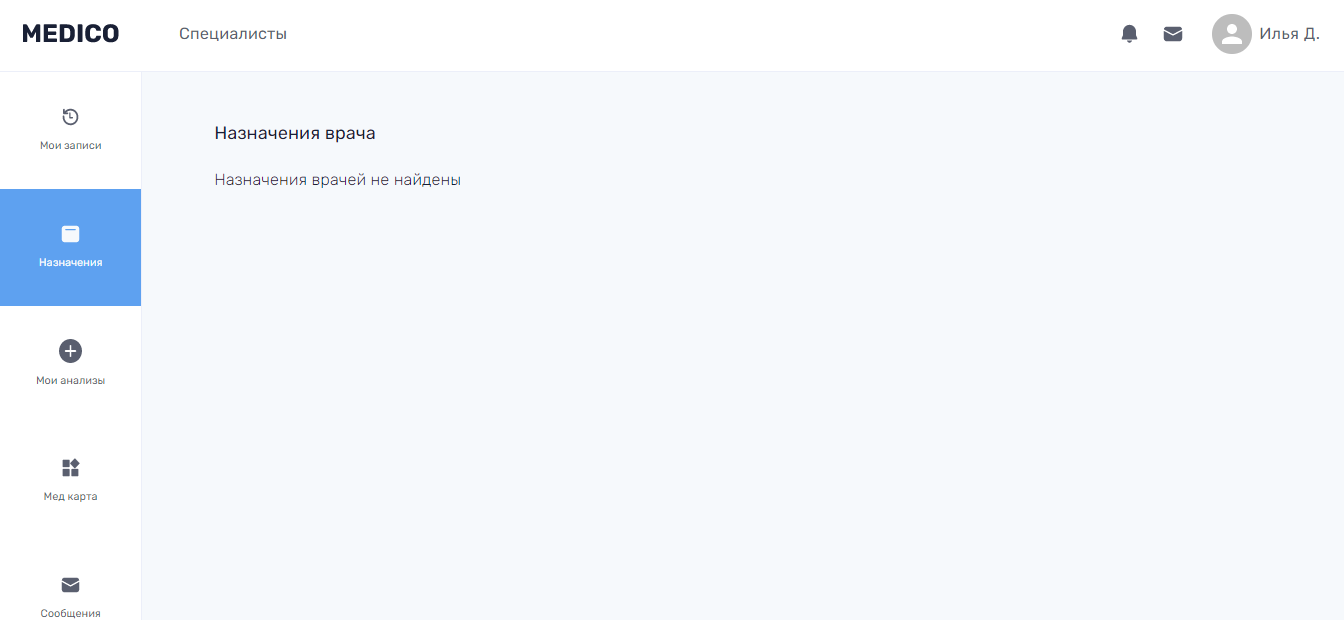


Рисунок 23 – Страница назначений

На странице “Мои анализы” пациент имеет возможность загрузить анализ либо снимок. Диалог загрузки файла, представленный на рисунке 24, вызывается при клике на кнопку, расположенную правее названия страницы.

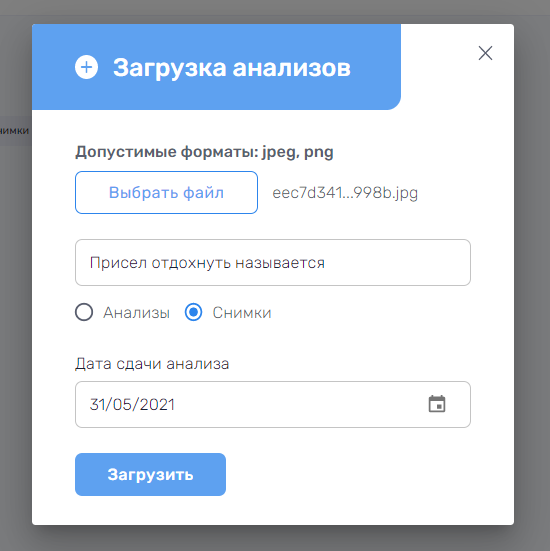


Рисунок 24 – Диалог загрузки анализов

После успешной загрузки файла на странице появится загружаемый анализ/снимок с подписью в виде даты сдачи и названия. Также все загружаемые файлы можно сортировать по их типу. Данная страница представлена на рисунке 25.

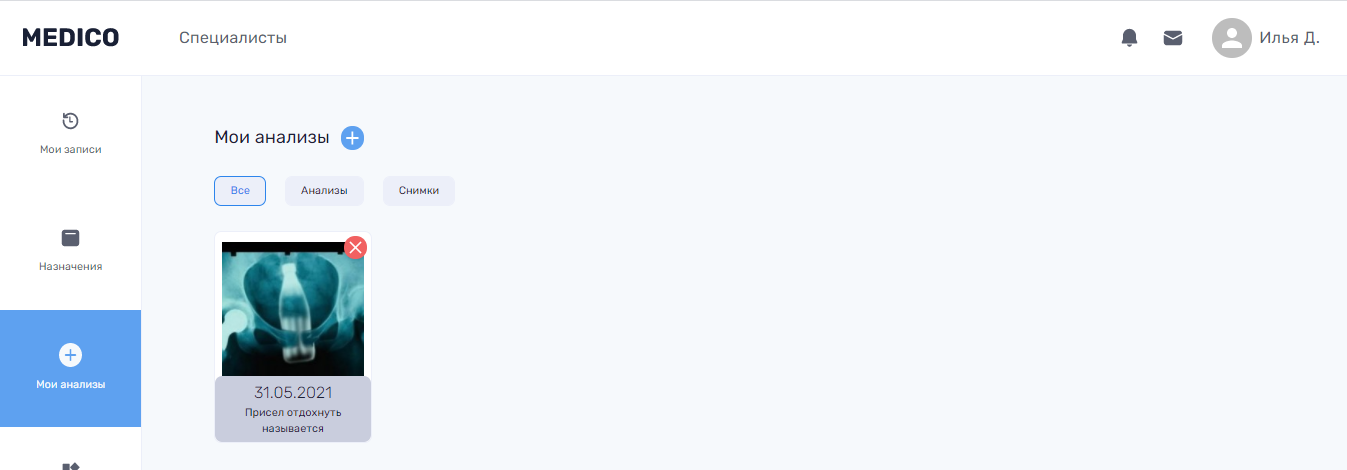


Рисунок 25 – Страница анализов

Также загружаемые файлы можно посмотреть более детально, кликнув по изображению анализа. Данная возможность представлена на рисунке 26.

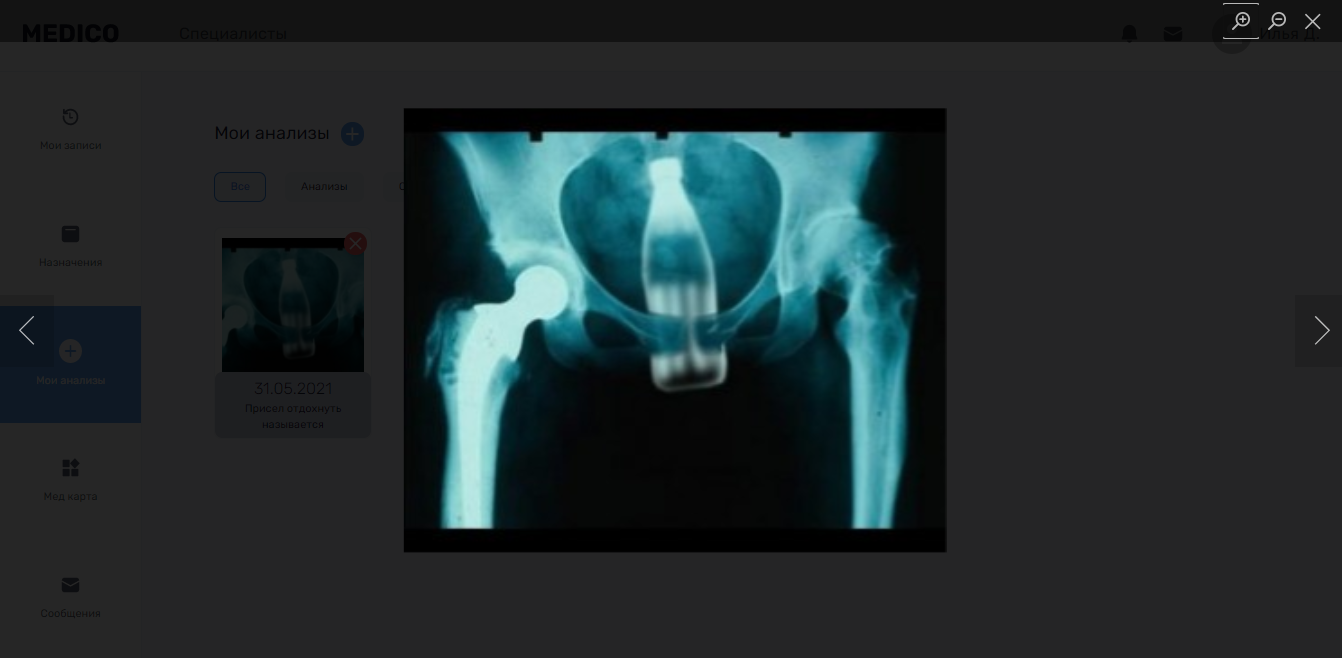


Рисунок 26 – Детальный просмотр анализа

Следующая страница “Мед. карта” предназначена для заполнения информации о себе, такой как: рост, вес, группа крови, аллергия, вредные привычки, переливание крови, операции и хронические заболевания. Также на данной странице автоматически считается Индекс Массы Тела с соответствующим пояснением. Данная страница представлена на рисунке 27.

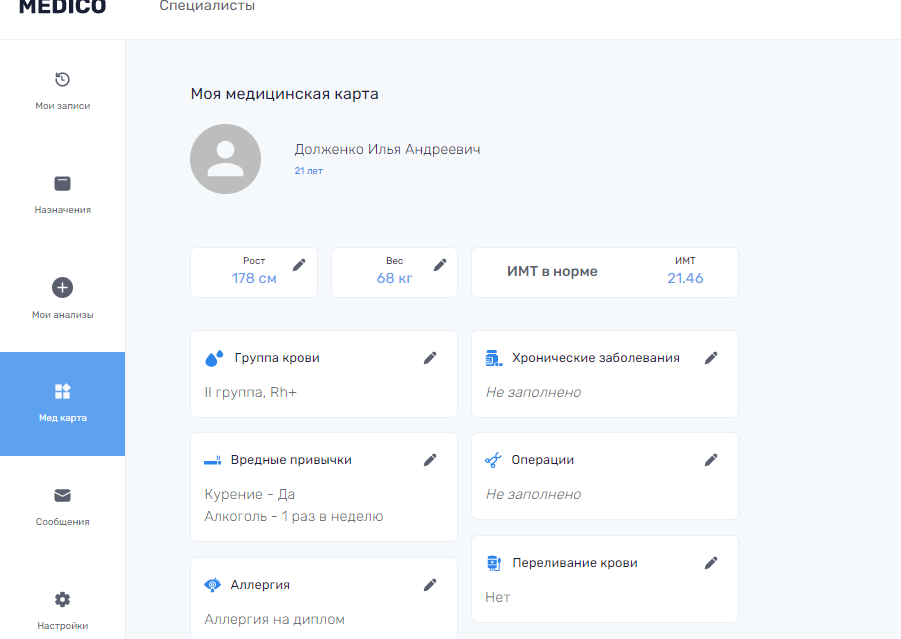


Рисунок 27 – Страница медицинской карты пациента

При клике на иконку редактирования открывается диалоговое окно с выбранными полями. Данный процесс представлен на рисунке 28.

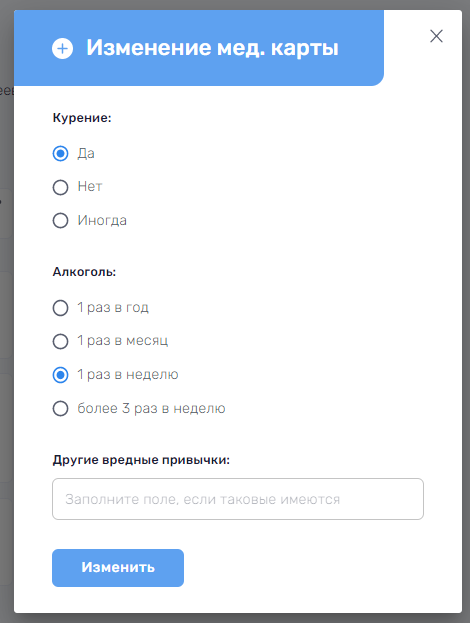


Рисунок 28 – Процесс изменения мед. Карты

Страница настроек является общей и доступна как для пациента, так и для врача. На данной странице, изображенной на рисунке 29, можно изменять как свое имя, фамилию, отчество, так и дату рождения, номер телефона, пол.

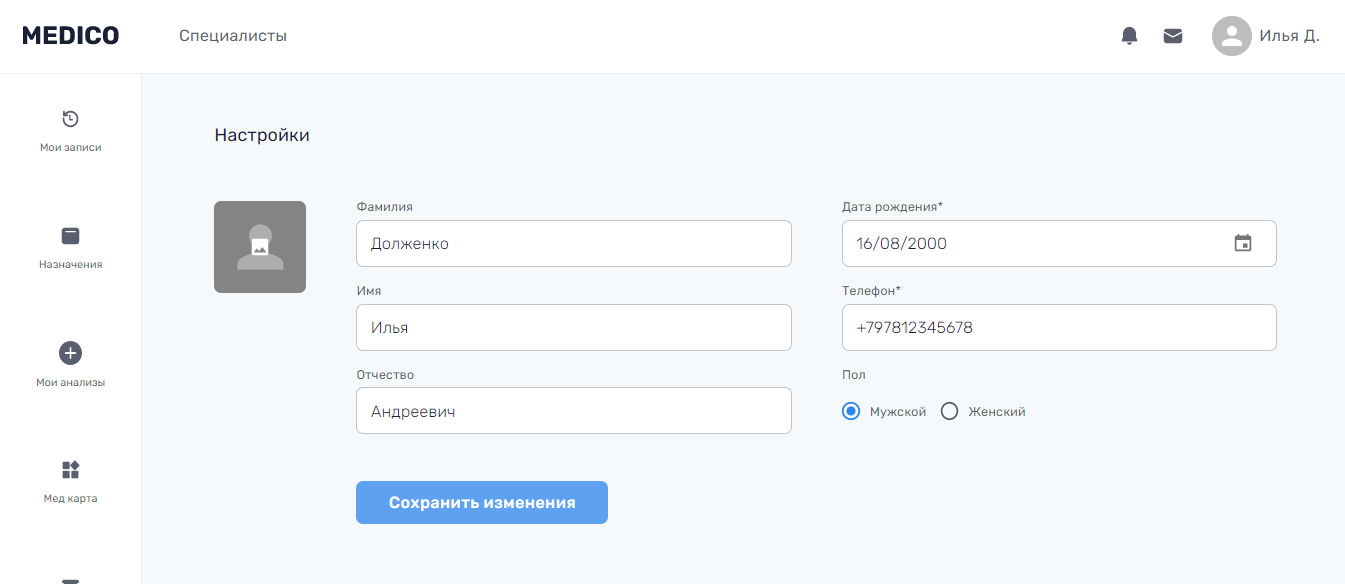


Рисунок 29 – Общая страница настроек

В отличие от пациента, личный кабинет врача имеет следующие страницы: главная, график работы и пациенты.

На главной странице личного кабинета врача находится информация о своем профиле. Имеется возможность просмотра отзывов и их подгрузка, просмотр и изменение образования и опыта работы, просмотр основных специальностей, выбранных при заполнении заявки на врача. Данный профиль врача представлен на рисунке 30.

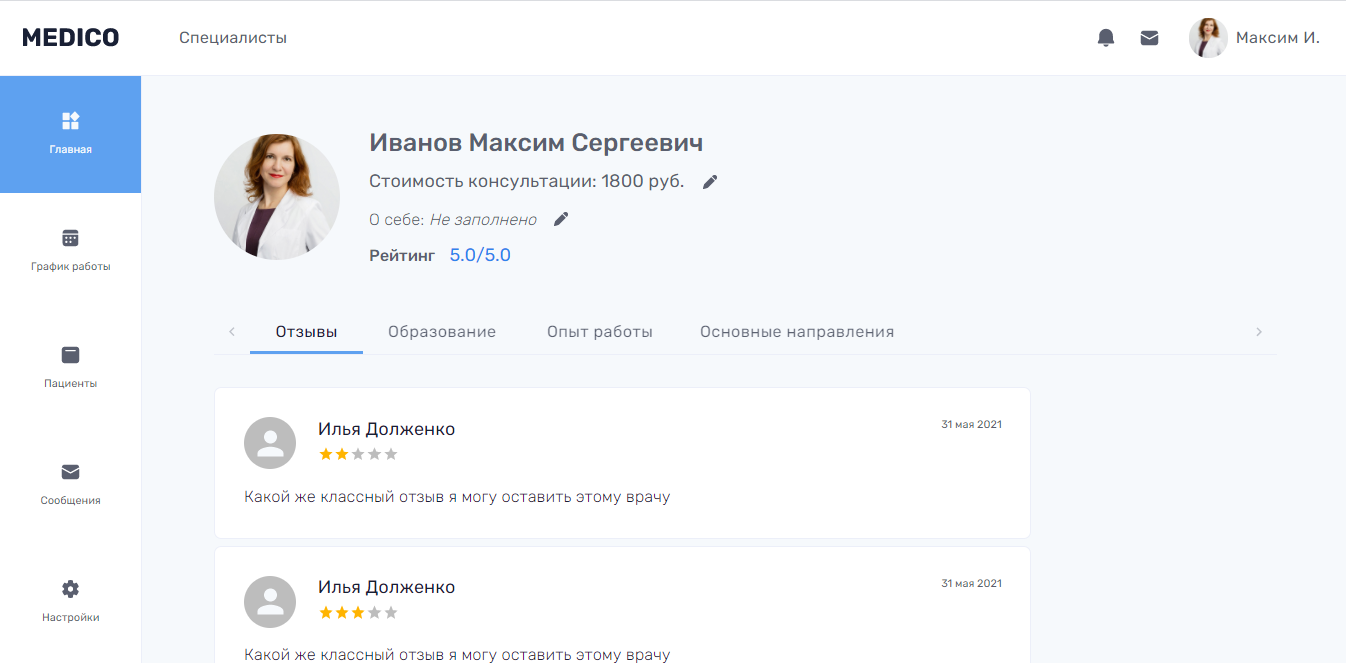


Рисунок 30 – Главная страница врача

При нажатии на кнопку редактирования стоимости консультации или информации о себе появляется диалоговое окно с соответствующей формой. Процесс редактирования информации о себе представлено на рисунке 31.



Рисунок 31 – Форма редактирования “О себе”

Редактирование образования и опыта работы происходит по аналогичному сценарию. Только теперь они содержат динамическое количество записей, т.е. имеется возможность добавить в одной форме сразу несколько записей. Пример редактирования образования приведен на рисунке 32.

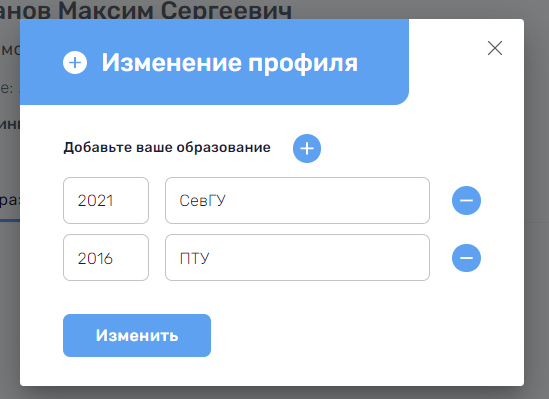


Рисунок 32 – Пример редактирования образования

На следующей странице “График работы” (рисунок 33) врач может изменить время своего приема по дням, либо установить дни без приема.

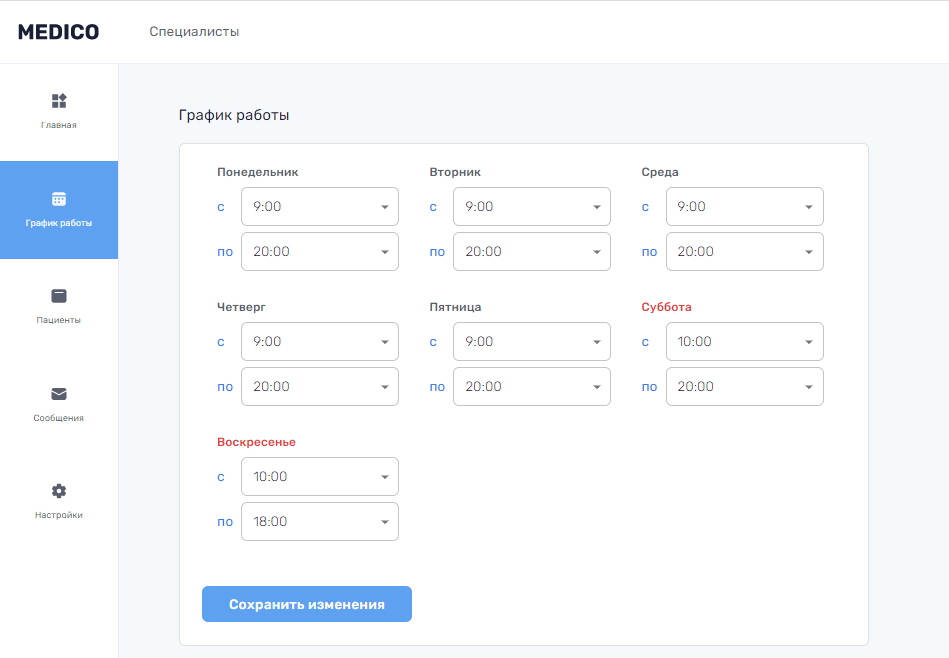


Рисунок 33 – Страница редактирования графика работы

Страница “Пациенты” (рисунок 34) разделена на два блока: список с пациентами по категории (новые или история) и календарь с выбором даты, по которой необходимо отображать пациентов.

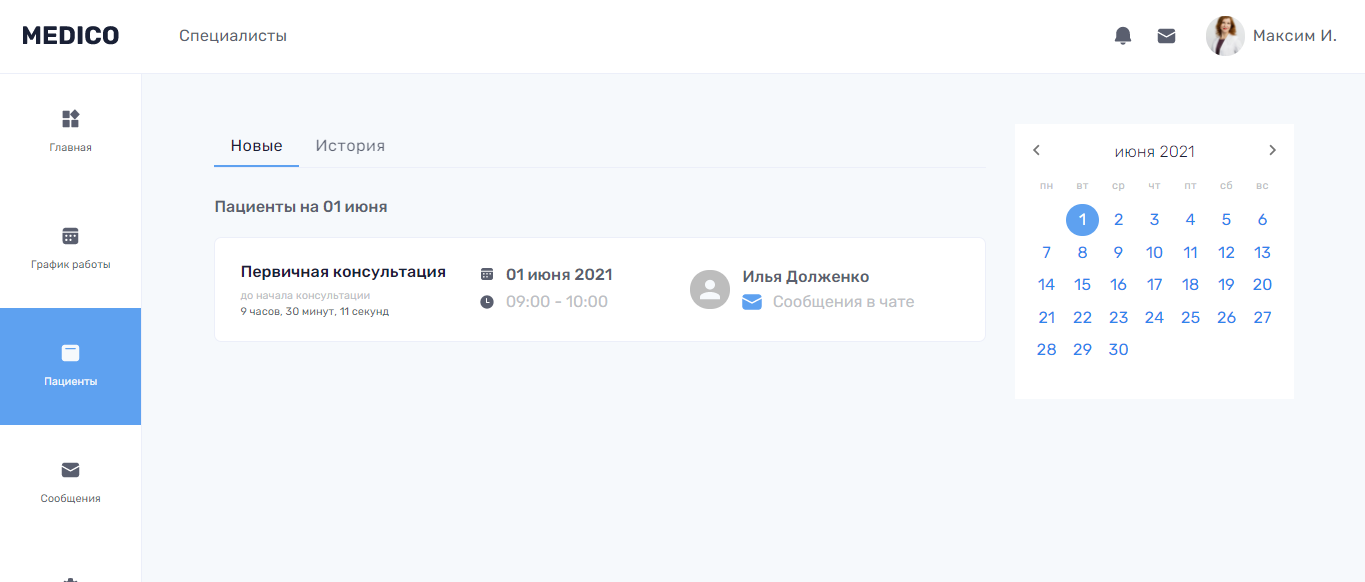


Рисунок 34 – Страница просмотра пациентов

При клике на пациента открывается страница просмотра его профиля, которая содержит его медицинскую карту, симптомы, анализы и прошлые назначения врачей. Страница профиля пациента представлена на рисунке 35.

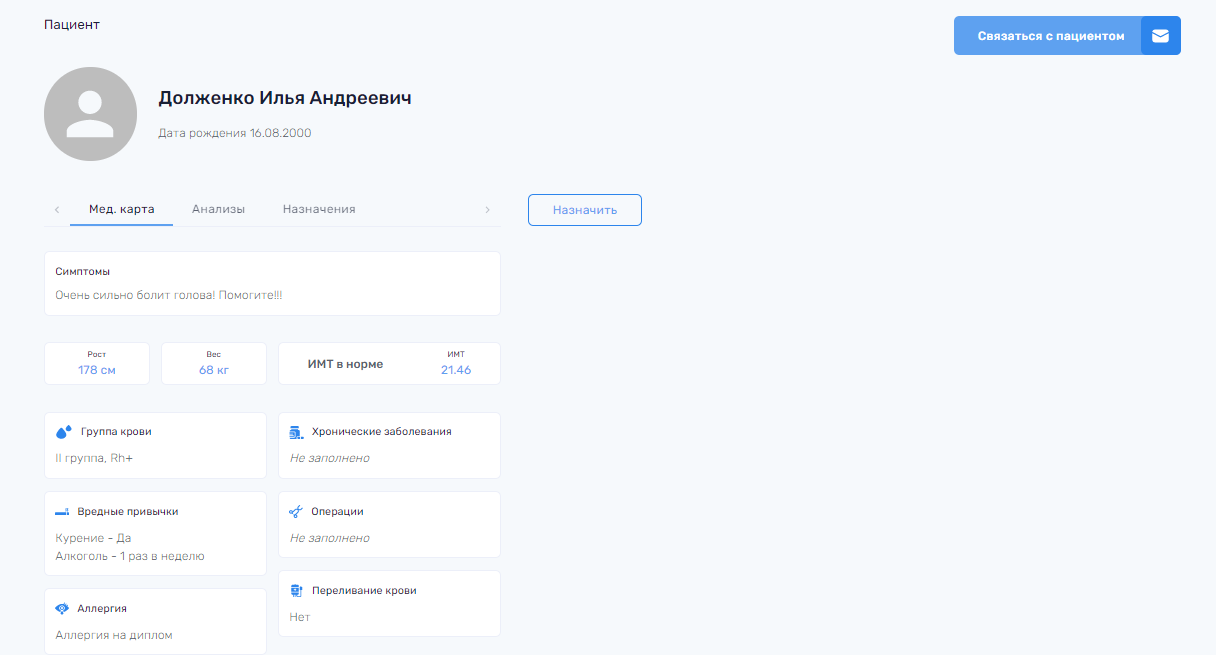


Рисунок 35 – Страница просмотра профиля пациента

На данной странице также находится кнопка назначения рекомендаций, при клике на которую открывается диалоговое окно с текстовым полем (рисунок 36).

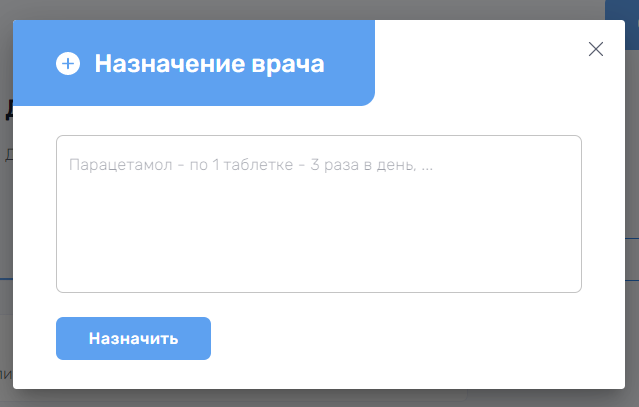


Рисунок 36 – Окно назначения рекомендаций пациенту

Также на странице находится кнопка “связаться с пациентом”, при клике на которую открывается диалог на общей странице чата. Страница “Сообщения” с выбранным диалогом изображена на рисунке 37.



Рисунок 37 – Страница диалога с врачом

В случае если диалог не выбран, то на странице отображается анимированное изображение с текстом “Выберите диалог”. Данная ситуация представлена на рисунке 38.

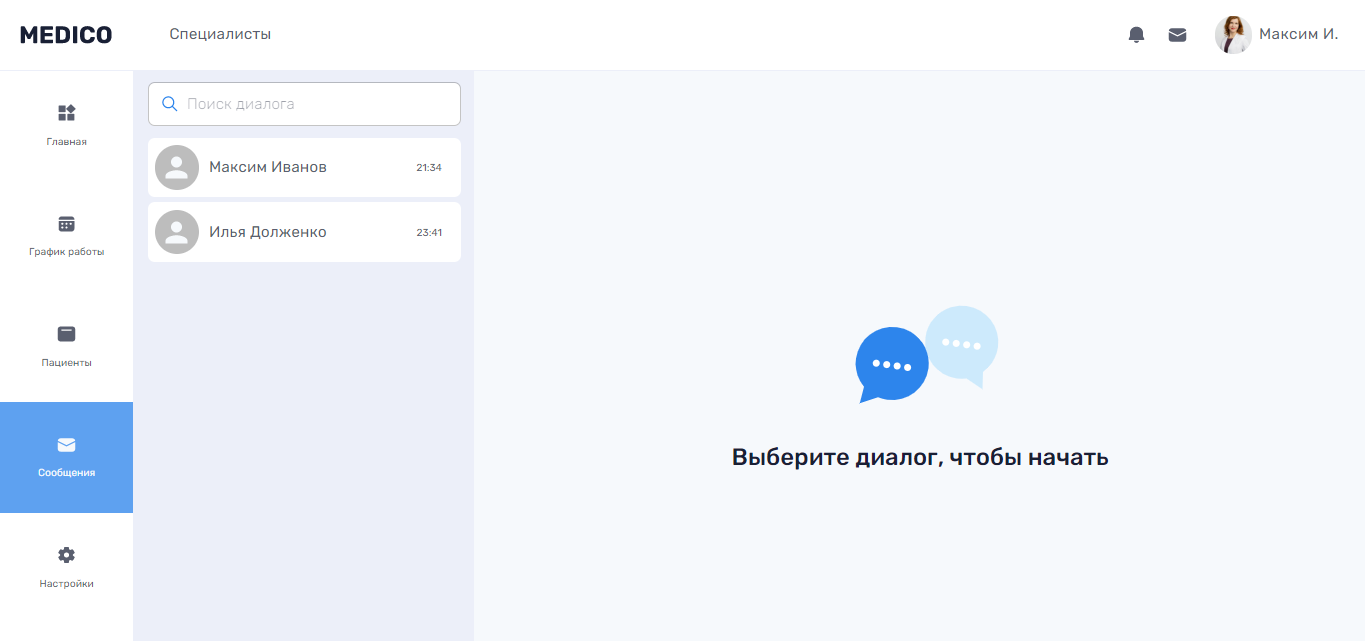


Рисунок 38 – Страница сообщений с невыбранным диалогом

Та же страница, но уже со стороны пациента имеет некоторое отличие – под именем врача в диалогах и верхней части сообщений написаны специальности, которыми обладает врач. Данный вид страницы представлен на рисунке 39.

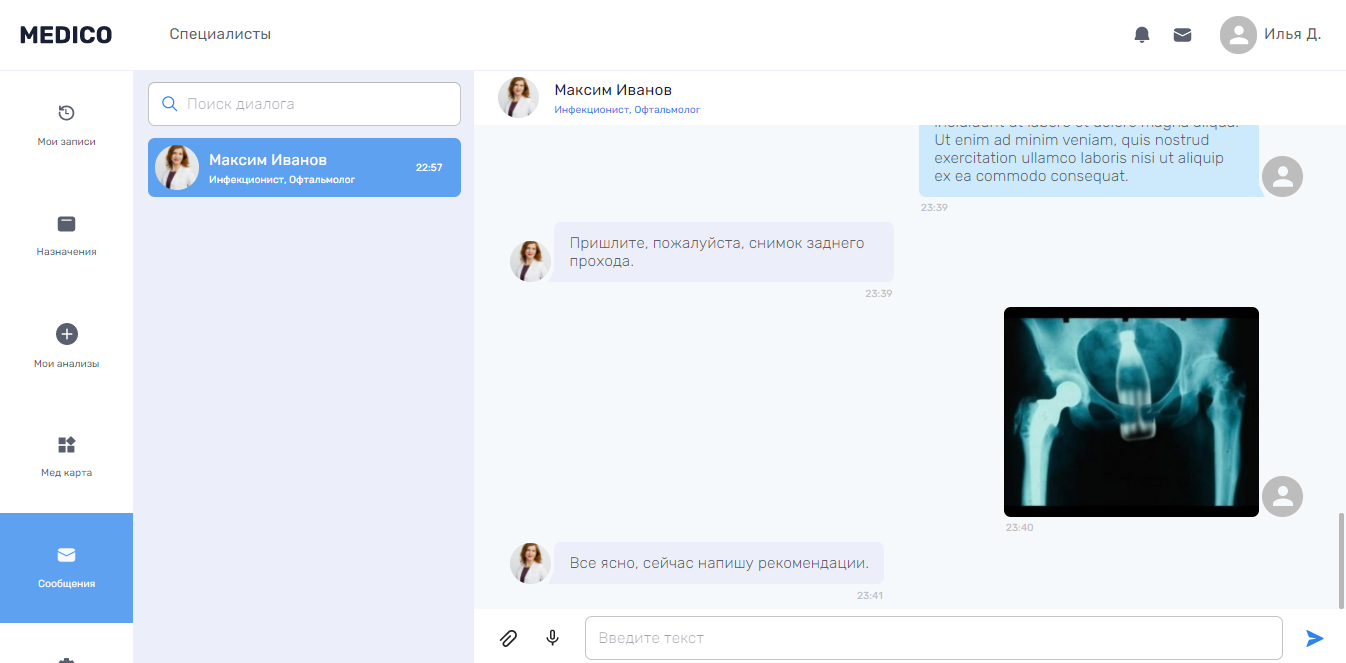


Рисунок 39 – Страница сообщений со стороны пациента

При общении в чате как пациент, так и врач имеют возможность отправить:

* фото (рисунок 40);

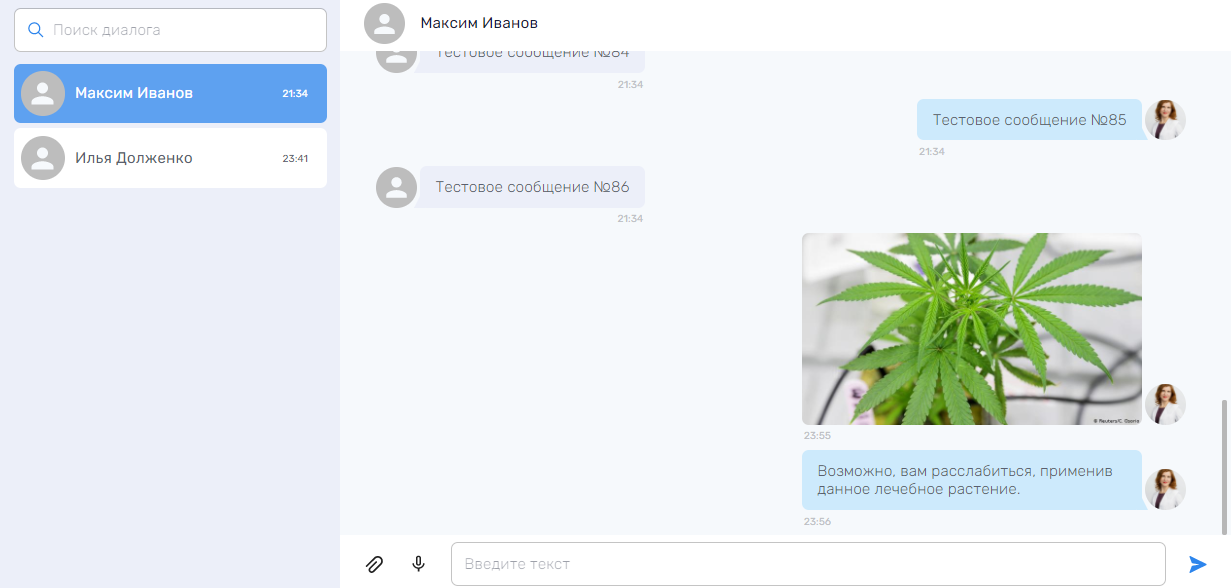


Рисунок 40 – Пример отправки изображения в чат

* аудиосообщение (рисунок 41);

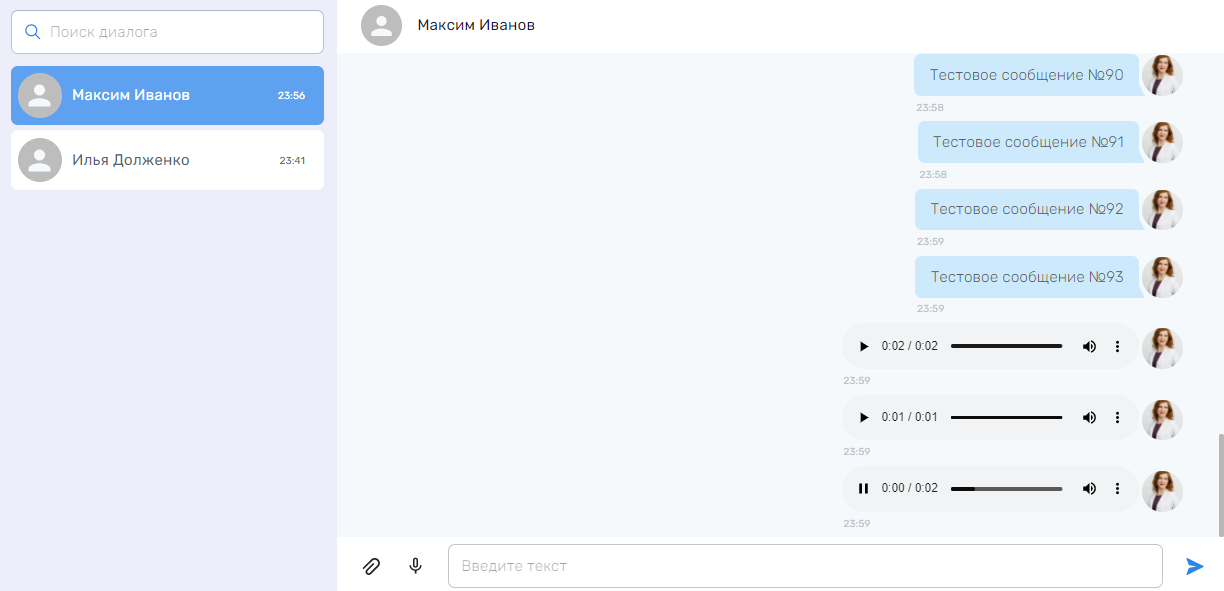


Рисунок 41 – Пример отправки аудиосообщения

* просто текстовый файл (рисунок 42).



Рисунок 42 – Пример отправки файла

**4 ТЕСТИРОВАНИЕ РАЗРАБОТАННОГО ПРОГРАММНОГО МОДУЛЯ «КЛИЕНТСКАЯ ЧАСТЬ ДЛЯ СИСТЕМЫ ОНЛАЙН-КОНСУЛЬТИРОВАНИЯ С ВРАЧАМИ»**

**4.1 Сценарии тестирования программного модуля**

//