|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  [*Факультет социально-экономических и компьютерных наук*](https://perm.hse.ru/scs/) |
| Елизаров Дмитрий Алексеевич  **Разработка информационной системы для автоматизации работы медицинских учреждений**  *Проект*  студента образовательной программы  «Разработка информационных систем для бизнеса»  по направлению подготовки 38.03.05 Бизнес-информатика  Руководитель  Старший преподаватель кафедры ИТБ  С.А. Волков | |

Пермь, 2024 год

**Оглавление**

[Введение 2](#_Toc164285827)

[Глава 1. Анализ 4](#_Toc164285828)

[1.1 Анализ предметной области 4](#_Toc164285829)

[1.2 Анализ современного рынка МИС 4](#_Toc164285830)

[1.3 Постановка задачи 6](#_Toc164285831)

[Глава 2. Проектирование 7](#_Toc164285832)

[2.1 Описание ключевых сущностей системы 7](#_Toc164285833)

[2.2 Описание доступных действий в системе 8](#_Toc164285834)

[2.3 Описание функциональных блоков 9](#_Toc164285835)

[2.4 Описание типов пользователей системы 10](#_Toc164285836)

[2.5 Описание доступных интерфейсов системы 11](#_Toc164285837)

[2.6 Диаграмма интерфейсов 15](#_Toc164285838)

[2.7 Прототипы основных интерфейсов 15](#_Toc164285839)

[2.7.1 Страница входа 17](#_Toc164285840)

[2.7.2 Профиль пользователя 18](#_Toc164285841)

[2.7.3 Клиенты (пациенты) 19](#_Toc164285842)

[2.7.4 Добавление нового клиента 20](#_Toc164285843)

[2.7.5 Сотрудники (пользователи) 21](#_Toc164285844)

[2.7.6 Приёмы (осмотры) 22](#_Toc164285845)

[2.7.7 Добавление нового приёма 23](#_Toc164285846)

[2.8 Проектирование базы данных 24](#_Toc164285847)

[2.8.1 Приведение к перовой нормальной форме 24](#_Toc164285848)

[2.8.2 Приведение к второй нормальной форме 25](#_Toc164285849)

[2.8.3 Приведение к третьей нормальной форме 27](#_Toc164285850)

[2.8.4 ER-диаграмма базы данных 29](#_Toc164285851)

[Глава 3. Реализация 30](#_Toc164285852)

[3.1 Выбор технологий разработки 30](#_Toc164285853)

[3.1.1 Серверная часть веб приложения 30](#_Toc164285854)

[3.1.2 Клиентская часть веб приложения 31](#_Toc164285855)

[3.1.3 Среда разработки 32](#_Toc164285856)

[3.2 Разработка дизайн системы 33](#_Toc164285857)

[3.3 Разработка серверной части 34](#_Toc164285858)

[3.3.1 Создание моделей данных 34](#_Toc164285859)

[3.3.2 Конечные узлы системы и запросы к БД 39](#_Toc164285860)

[3.4 Разработка клиентской части 41](#_Toc164285861)

[Глава 4. Тестирование 47](#_Toc164285862)

[Глава 5. Развертывание 48](#_Toc164285863)

[Список литературы 49](#_Toc164285864)

# **Введение**

В современном медицинском сообществе доступ к актуальной и структурированной информации играет важную роль в обеспечении высокого уровня обслуживания пациентов и эффективного управления медицинскими организациями. В этом контексте создание и использование медицинских информационных систем (МИС) имеет большое значение.

Но есть ряд проблем, которые неблагоприятно влияют на пользовательский опыт. Во-первых, многие современные медицинские информационные системы имеют перегруженный и недружелюбный интерфейс, который затрудняет получение, обработку и анализ информации. Во-вторых, популярные медицинские системы имеют дорогостоящую ежемесячную подписку, плату за внедрение и поддержку, что в свою очередь ограничивает развитие малого бизнеса и препятствует совершенствованию данного вида продукта на национальном рынке.

Введем несколько проблемных вопросов:

1. Каким образом необходимо оптимизировать пользовательский интерфейс, чтобы исключить возможные проблемы с получением и обработкой информации?
2. Как обеспечить доступность медицинских информационных систем для малого бизнеса?
3. Какие практики следует применять для укрепления сферы отечественных медицинских информационных систем на национальном рынке?

Целью данного проекта является проектирование и разработка открытой медицинской информационной системы для оптимальной автоматизации работы поликлиник и частных клиник. Открытость системы обеспечивается бесплатным распространением программного продукта на базе open-source и возможностью модификации системы сторонними разработчиками.

Объектом в данном исследовании является подход к проектированию, разработке и распространению программного продукта.

Предметом изучения является медицинская информационная система с открытым исходным кодом, которая позволяет оптимально оптимизировать работу медицинских учреждений и предоставить качественный пользовательский опыт.

Определим основные задачи проекта:

1. Проанализировать предметную область и рынок медицинских информационных систем;
2. Спроектировать программную систему (обозначить функциональные требования);
3. Разработать программный продукт;
4. Провести тестирование серверной и клиентской части;
5. Подвести итоги проектирования и разработки информационной системы.

Выявление правильного подхода к проектированию и разработке медицинских информационных систем позволит максимизировать пользу для бизнеса и увеличить количество программных продуктов отечественного производства, что в свою очередь положительно повлияет на национальную сферу медицинских услуг.

# **Глава 1. Анализ**

В данной главе описывается анализ предметной области и рынка современных медицинских информационных решений для дальнейшего проектирования программной системы.

## **Анализ предметной области**

Медицинская информационная система (МИС) – это система автоматизации документооборота для медицинских учреждений, в которой объединены система поддержки принятия врачебных решений, электронные медицинские карты пациентов, данные медицинских исследований в цифровой форме, данные мониторинга состояния пациента с медицинских приборов, средства общения между сотрудниками, финансовая и административная информация [1].

В данном проекте было принято решение сократить данное определение, опустив некоторые системные возможности, чтобы упростить проектирование и ускорить разработку программного продукта.

Медицинские учреждения сталкиваются с проблемой неэффективного управления данными, ручное ведение медицинских записей приводит к ошибкам в записях, потере части данных и увеличивает время на поиск информации.

Современная медицинская информационная система должна быть безопасной, легко масштабируемой и удобной в использовании. Важно обеспечить доступный пользовательский интерфейс для разного уровня квалификации сотрудников.

Медицинская информация является строго конфиденциальной, поэтому информационная система должна предоставлять инструменты защиты чувствительных данных, такие как шифрование, механизмы авторизации и аутентификации пользователей системы.

## **Анализ современного рынка МИС**

Для корректного составления требований к разрабатываемой информационной системе, необходимо проанализировать существующие решения, выделить их основные преимущества и недостатки, провести результирующее сравнение. Для проведения сравнения системы было выбрано 3 наиболее популярных решения: Медиалог, 1С:Медицина и qMS [2, 3, 4].

Подробный анализ и сравнение данных информационных систем представлен в Таблице 1.

Таблица 1 – Достоинства и недостатки существующих решений

|  |  |  |
| --- | --- | --- |
| **МИС** | **Достоинства** | **Недостатки** |
| Медиалог | Интерфейс и навигация для пользователей разного уровня квалификации;  Возможность интеграции с сторонними медицинскими системами (ЛИС, ПАКС);  Учёт пациентов, ведение медицинских карт и истории болезни, отчётность и аналитика. | Распространяется на платной основе, может быть недоступна для небольших медицинских учреждений или частных практик;  Требует большого количества времени на внедрение и обучение сотрудников;  Предоставляет устаревший формат интерфейса; |
| 1С:Медицина | Предоставляет возможность интеграции со сторонними продуктами компании 1С, поддерживает стандарты отчётности;  Учёт пациентов, ведение медицинских карт и истории болезни, управление финансами и складским отчётом;  Простой пользовательский интерфейс и сквозная навигация; | Не предоставляет возможность модификации системы под специфические потребности конкретного учреждения;  Требует большого количества времени на внедрение и обучение сотрудников;  Сложность поддержки и обновления системы в долгосрочной перспективе; |
| qMS | Имеет узкую специализацию, направленную на управление качеством и безопасностью;  Инструменты автоматизации процессов управления качеством, мониторинга и аналитики;  Доступна для малого бизнеса из-за фиксации на конкретных аспектах; | Отсутствуют функции, необходимые для полноценного управления медицинским учреждением;  Предоставляет устаревший формат интерфейса; |

Исходя из результатов сравнения можно сделать вывод, что данные системы схожи в своих недостатках, можно выделить три основные проблемы: распространение на платной основе (возможная недоступность для малого бизнеса и частных практик), проприетарность (закрытость) системы, устаревший пользовательский интерфейс, препятствующий легкому внедрению и обучению (перегруженность и излишество).

Реализуемая система должна предоставлять удобный и интуитивно понятный пользовательский интерфейс, должна покрывать обширную функциональную область и распространяться на базе открытого исходного кода (open-source).

## **Постановка задачи**

Для успешного завершения проектирования и разработки медицинской информационной системы необходимо разбить проект на смысловые блоки и выделить основные задачи:

1. Спроектировать программную систему (обозначить функциональные требования):
   1. Описать ключевые сущности системы;
   2. Описать доступные действия в системе;
   3. Описать функциональные блоки;
   4. Описать систему ролей пользователей;
   5. Описать доступные интерфейсы системы;
   6. Описать диаграмму интерфейсов;
   7. Собрать прототипы интерфейсов;
   8. Спроектировать базу данных.
2. Разработать программный продукт
   1. Разработать серверную часть (back-end разработка);
   2. Разработать дизайн-систему;
   3. Разработать клиентскую часть (front-end разработка).
3. Провести тестирование серверной и клиентской части;
4. Развернуть веб-приложение в сети Интернет с публичным доступом.

Разбиение процесса проектирования и разработки позволит выстроить структуру проекта и четко обозначить задачи, которые должны быть реализованы в финальной версии программного продукта.

# **Глава 2. Проектирование**

Перед началом разработки информационной системы необходимо провести этап проектирования программного продукта, чтобы выявить возможности и избежать лишних временных затрат. В данной главе описаны основные функциональные требования к разрабатываемой системе.

## **Описание ключевых сущностей системы**

Исходя из анализа предметной области, необходимо описание основных сущностей медицинской информационной системы, что в свою очередь является ключевым шагом для полного понимания её структуры и необходимой функциональности.

**Пользователь** – конкретная учётная запись со своими реквизитами входа в систему. В данной случае под пользователями подразумеваются сотрудники компании, в которой происходит внедрение информационной системы (врачи, администраторы, отдел кадров и т.п.).

**Роль** - это именованный набор прав или разрешений, которым наделяются пользователи, принадлежащие к этой роли. Для упрощения разработки было принято решение сократить и обобщить возможный персонал до трёх ролей:

* врач (занимается приёмом пациентов, ведет их личную историю болезни);
* администратор (имеет полный доступ к защищенным интерфейсам системы);
* гость (пользователь без роли).

**Отдел** - направление рабочей деятельности (администрация, врачебный отдел, отдел кадров и т.п.).

**Должность** - конкретный вид рабочей деятельности в конкретном отделе.

**Категория квалификации** – показатель, отражающий степень компетентности конкретного врача. Существует три основные категории:

* вторая категория (стаж – не менее 3 лет);
* первая категория (стаж – не менее 5 лет);
* высшая категория (стаж – не менее 7 лет).

**Пациент –** конкретный клиент компании.

**Приём (осмотр) –** процедура, при которой медицинский работник проводит проверку здоровья пациента.

**Статус приема** – показатель текущего состояния процесса приёма пациента у конкретного врача. В данном случае статус может принимать 4 различных значений:

* запланирован (осмотр ещё не начался);
* в процессе (врач проводит осмотр);
* завершен (осмотр пациента успешно завершен);
* отменен (осмотр пациента был отменен).

**История болезни –** это набор, история развития/спада болезни, относится к конкретному пациенту.

## **Описание доступных действий в системе**

Пользователи МИС могут наделяться определенным набором прав, позволяющим осуществлять следующие действия:

Аутентификация – вход в личный кабинет (профиль) пользователя.

Просмотр списка клиентов – если роль сотрудника определена как «врач», то он может просмотреть список всех своих клиентов.

Редактирование, создание и удаление карточки пациента. Ввод информации о новых пациентах, включая ФИО, контактные данные, дату рождения и другие идентификационные данные.

Просмотр списка сотрудников – пользователь может при необходимости найти информацию про другого сотрудника, например, если ему необходимо связаться с ним.

Редактирование, создание и удаление карточки сотрудника – если роль пользователя определена как «администратор», то он может модифицировать список сотрудников: добавлять, изменять и удалять пользователей. Ввод информации о новых сотрудниках, включая ФИО, контактные данные, дату рождения и другие идентификационные данные.

Редактирование, создание и удаление приёмов - запись на приём к врачу или другому медицинскому специалисту через интерфейс МИС. Это может включать в себя выбор даты и времени приёма, а также специализацию врача. Запись на приём осуществляется исключительно сотрудниками компании. Редактирование, создание и удаление статуса сделок.

## **Описание функциональных блоков**

Опишем функциональные блоки медицинской информационной системы, которые являются основными компонентами системы и определяют её функциональные возможности.

**Аутентификация –** логика входа в личный кабинет. К данному функциональному блоку относятся все процедуры и модули, отвечающие за вход пользователя в личный кабинет системы.

**Авторизация –** система управления ролями пользователей и связки определенных ролей и определенных пользователей. К данному функциональному блоку относятся модули системы, отвечающие за проверку и раздачу ролей пользователям.

**Просмотр пользователей** – просмотр списка пользователей. К данному функциональному блоку относятся модули системы, отвечающие за получение основной информации о всех существующих пользователях системы.

**Редактирование пользователей.** К данному функциональному блоку относятся модули системы, отвечающие за редактирование карточек пользователей системы, если пользователь имеет роль «администратор».

**Просмотр карточки пользователей** – просмотр карточки конкретного пользователя системы. К данному функциональному блоку относятся модули системы, отвечающие за получение основной информации конкретном существующем пользователе.

**Просмотр клиентов** – просмотр списка всех клиентов. К данному функциональному блоку относятся модули системы, отвечающие за получение основной информации о всех существующих пользователях системы.

**Редактирование клиентов.** К данному функциональному блоку относятся модули системы, отвечающие за редактирование карточек существующих клиентов системы, если пользователь имеет роль «администратор».

**Просмотр карточки клиента** – просмотр карточки конкретного клиента системы. К данному функциональному блоку относятся модули системы, отвечающие за получение основной информации о конкретном существующем пользователе.

**Просмотр приёмов** – просмотр списка всех осмотров. К данному функциональному блоку относятся модули системы, отвечающие за получение основной информации о всех осмотрах конкретного пользователя системы.

**Редактирование приёмов.** К данному функциональному блоку относятся модули системы, отвечающие за редактирование карточек существующих осмотров конкретного пользователя системы, если пользователь имеет роль «врач».

**Просмотр карточки приёма** – просмотр карточки конкретного осмотра конкретного пользователя системы. К данному функциональному блоку относятся модули системы, отвечающие за получение основной информации о конкретном существующем осмотре конкретного пользователя системы.

## **Описание типов пользователей системы**

Типы пользователей - это набор стартовых ролей в системе, необходимых для начального функционирования. Система включает в себя три возможных типа пользователей:

* гость (анонимный пользователь) – не имеет права входа в личный кабинет пользователя, не может использовать представленный в защищенном разделе сайта функционал, не может обращаться к внутренним интерфейсам системы;
* администратор (пользователь с неограниченными правами) – может входить в личный кабинет при помощи своей учетной записи, может обращаться к любым интерфейсам системы и использовать весь представленный в системе функционал;
* врач (сотрудник компании владельца) – может входить в личный кабинет при помощи своей учетной записи, наделяется определенным набором прав для управления (создания/редактирования/удаления) отдельными сущностями системы.

Очень важно обеспечить невозможность удаления или изменения данного набора стартовых ролей системы, это может привести к потере данных без возможности восстановления.

## **Описание доступных интерфейсов системы**

Интерфейсы системы играют ключевую роль в удобстве использования, эффективности работы и удовлетворении потребностей пользователя, они обеспечивают взаимодействие пользователей с системой и доступ к её данным и функциональности. Опишем доступные интерфейсы разрабатываемой медицинской информационной системы:

**Страница входа –** страница ввода логина и пароля для входа в защищенную часть сайта (личный кабинет). Данный интерфейс содержит следующие элементы и блоки: логотип компании, форма ввода логина и пароля.

**Профиль пользователя** – защищенная часть сайта. Страница с информацией текущего пользователя. Данный интерфейс содержит следующие элементы и блоки:

* фото профиля;
* ФИО сотрудника;
* отдел;
* должность;
* паспортные данные;
* форма для смены пароля.

**Панель управления** – боковое навигационное меню. Данный интерфейс подразумевает центральный интерфейс для пользователя в авторизованном состоянии и предполагает наличие визуальных карточек для перехода к прочим внутренним интерфейсам системы. Реализует сквозную навигацию.

**Клиенты (пациенты) –** страница со всеми клиентами конкретного пользователя. Данный интерфейс предоставляет следующие возможности:

* 1. Добавление клиента;
  2. Редактирование клиента;
  3. Просмотр карточки клиента.

Данный интерфейс содержит следующие элементы и блоки:

* таблица со списком клиентов;
* счётчик общего количества клиентов;
* строка поиска;
* кнопка добавления нового клиента;
* кнопка просмотра карточки клиента;
* пагинация;
* форма для создания нового клиента;
* элемент карточки клиента.

**Карточка клиента (пациента) –** отдельный элемент, который содержит подробную информацию о конкретном клиенте. Данный интерфейс содержит следующие элементы и блоки:

* ФИО клиента;
* дата рождения;
* личная почта;
* личный телефон;
* номер полиса ОМС;
* дата/время открытия и закрытия истории болезни;
* паспортные данные.

**Сотрудники (пользователи и роли) -** страница со всеми пользователями системы. Данный интерфейс предоставляет следующие возможности:

* 1. Добавление сотрудника;
  2. Редактирование сотрудника;
  3. Просмотр карточки сотрудника.

Данный интерфейс содержит следующие элементы и блоки:

* таблица со списком сотрудников;
* счётчик общего количества сотрудников;
* строка поиска;
* кнопка добавления нового сотрудника;
* кнопка просмотра карточки сотрудника;
* пагинация;
* форма для создания нового пользователя;
* элемент карточки пользователя.

**Карточка пользователя (сотрудника) –** отдельный элемент, который содержит подробную информацию о конкретном пользователе. Данный интерфейс содержит следующие элементы и блоки:

* фото профиля;
* ФИО пользователя;
* номер лицензии;
* корпоративная почта;
* телефон;
* отдел;
* должность;
* стаж;
* категория квалификации.

**Приёмы -** страница со всеми приёмами текущего пользователя. Данный интерфейс предоставляет следующие возможности:

* 1. Добавление осмотра;
  2. Редактирование осмотра;
  3. Просмотр карточки осмотра.

Данный интерфейс содержит следующие элементы и блоки:

* таблица со списком приёмов;
* счётчик общего количества осмотров;
* строка поиска;
* кнопка добавления нового осмотра;
* кнопка просмотра карточки осмотра;
* пагинация;
* форма для создания нового осмотра;
* элемент карточки осмотра.

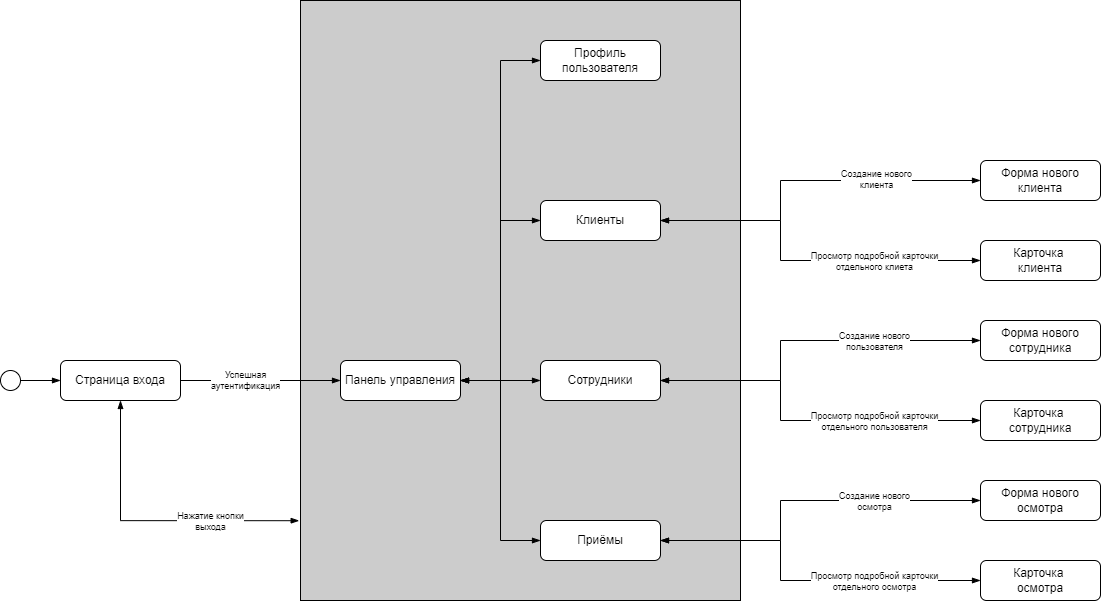
**Карточка осмотра –** отдельный элемент, который содержит подробную информацию о осмотре текущего пользователя. Данный интерфейс содержит следующие элементы и блоки:

* дата/время начала осмотра;
* время окончания осмотра;
* цель визита;
* диагноз;
* назначения;
* ФИО сотрудника;
* ФИО клиента;
* статус осмотра.

Необходимо уточнить, что в зависимости от наличия прав у текущего пользователя, набор элементов для редактирования/удаления и просмотра информации внутри каждого интерфейса может меняться.

## **Диаграмма интерфейсов**

В качестве среды проектирования была выбрана бесплатная программа на основе открытого исходного кода Drawio. На Рисунке 1 представлена диаграмма интерфейсов разрабатываемой медицинской информационной системы.



***Рисунок 1 – Диаграмма интерфейсов***

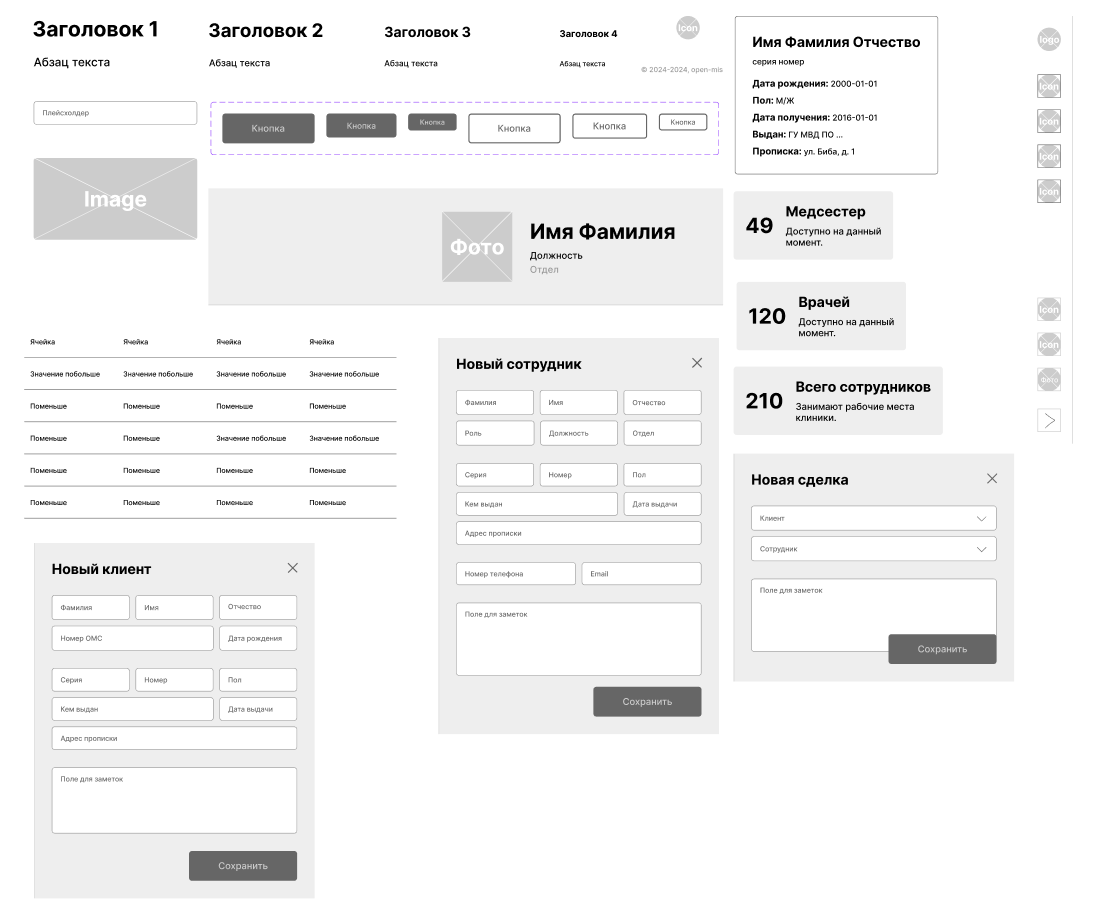
Для эффективного использования системы необходимо обеспечить сквозную навигацию (обозначено серой зоной), данный функционал реализует интерфейс «Панель управления». При помощи данного интерфейса возможен взаимный переходи из одного основного места системы в другое.

## **Прототипы основных интерфейсов**

В качестве среды проектирования прототипов интерфейсов был выбран онлайн-сервис для разработки интерфейсов и прототипирования – «Figma». Данное решение было выбрано в качестве основного инструмента прототипирования по следующим причинам:

* возможность бесплатного использования;
* удобство пользовательского интерфейса;
* популярность данного решения;
* наличие личного опыта использования инструмента.

Перед созданием прототипов интерфейсов необходимо создать набор основных компонентов, из которых в дальнейшем будут собираться макеты прототипов. На Рисунке 2 представлены основные компоненты сборки интерфейсов.

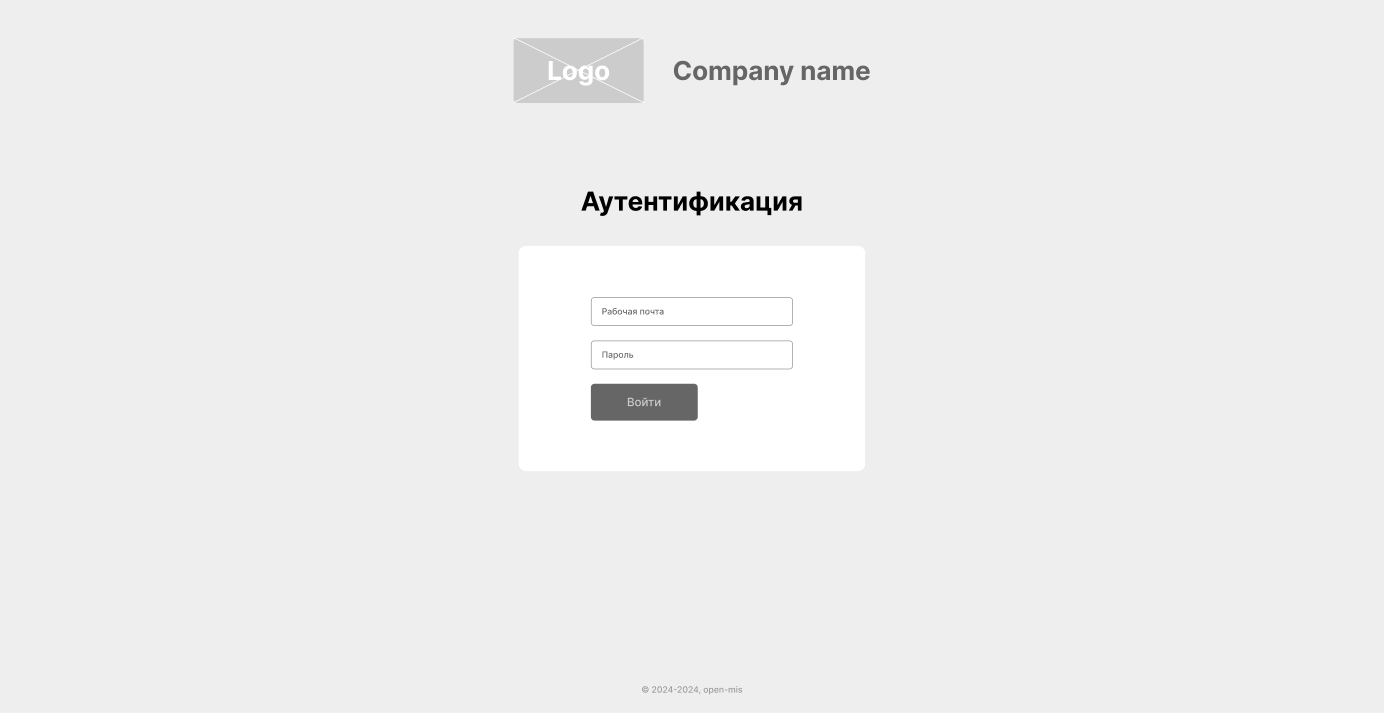


***Рисунок 2 – готовые компоненты***

Далее в данном разделе будут представлены прототипы основных интерфейсов разрабатываемой информационной системы.

**2.7.1 Страница входа**

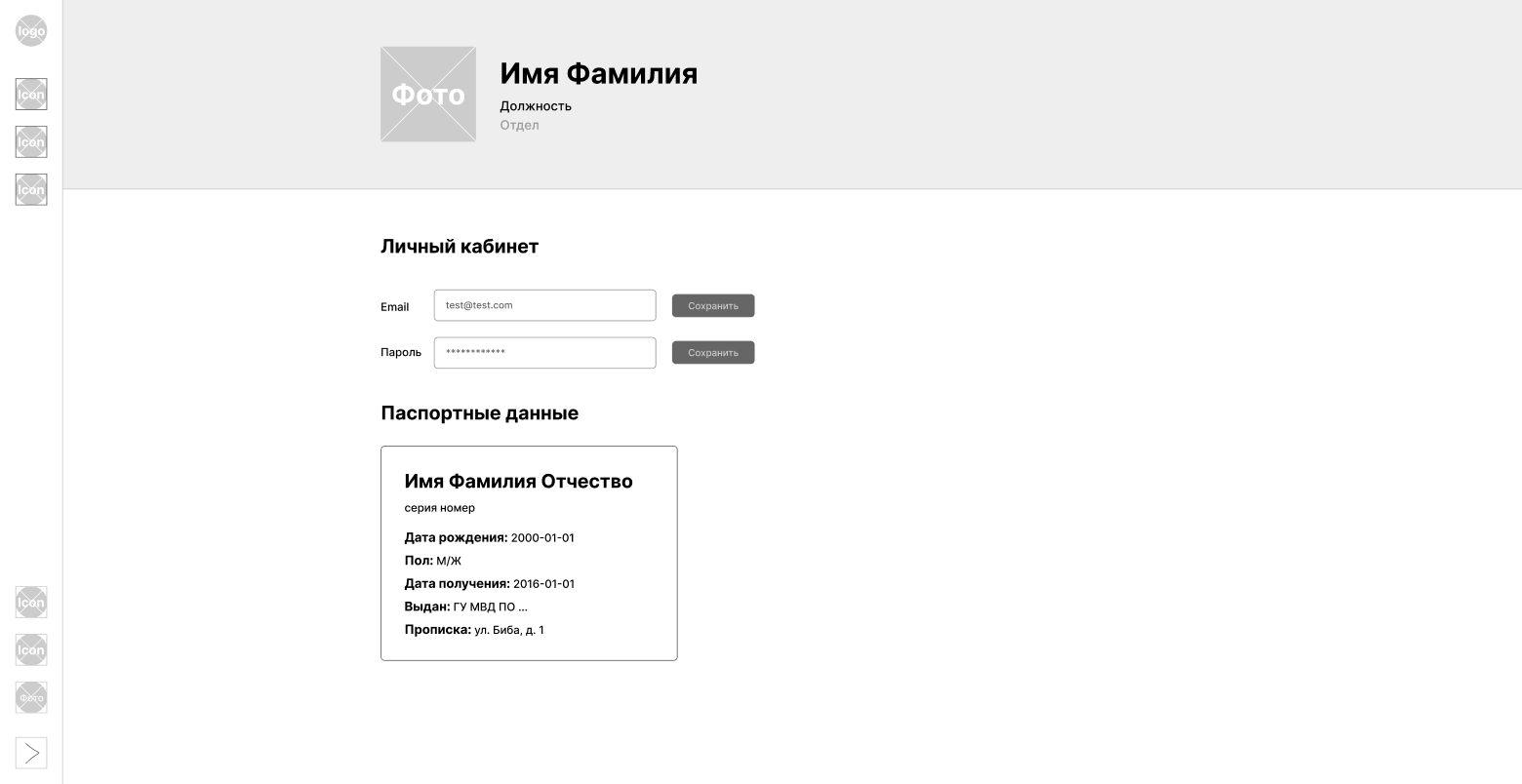
Данный интерфейс состоит из логотипа компании и формы для предоставления информации о текущем пользователе, реализует механизм аутентификации пользователя в системе. Прототип интерфейса «Страница входа» представлен на Рисунке 3.



***Рисунок 3 – Прототип страницы «Страница входа»***

**2.7.2 Профиль пользователя**

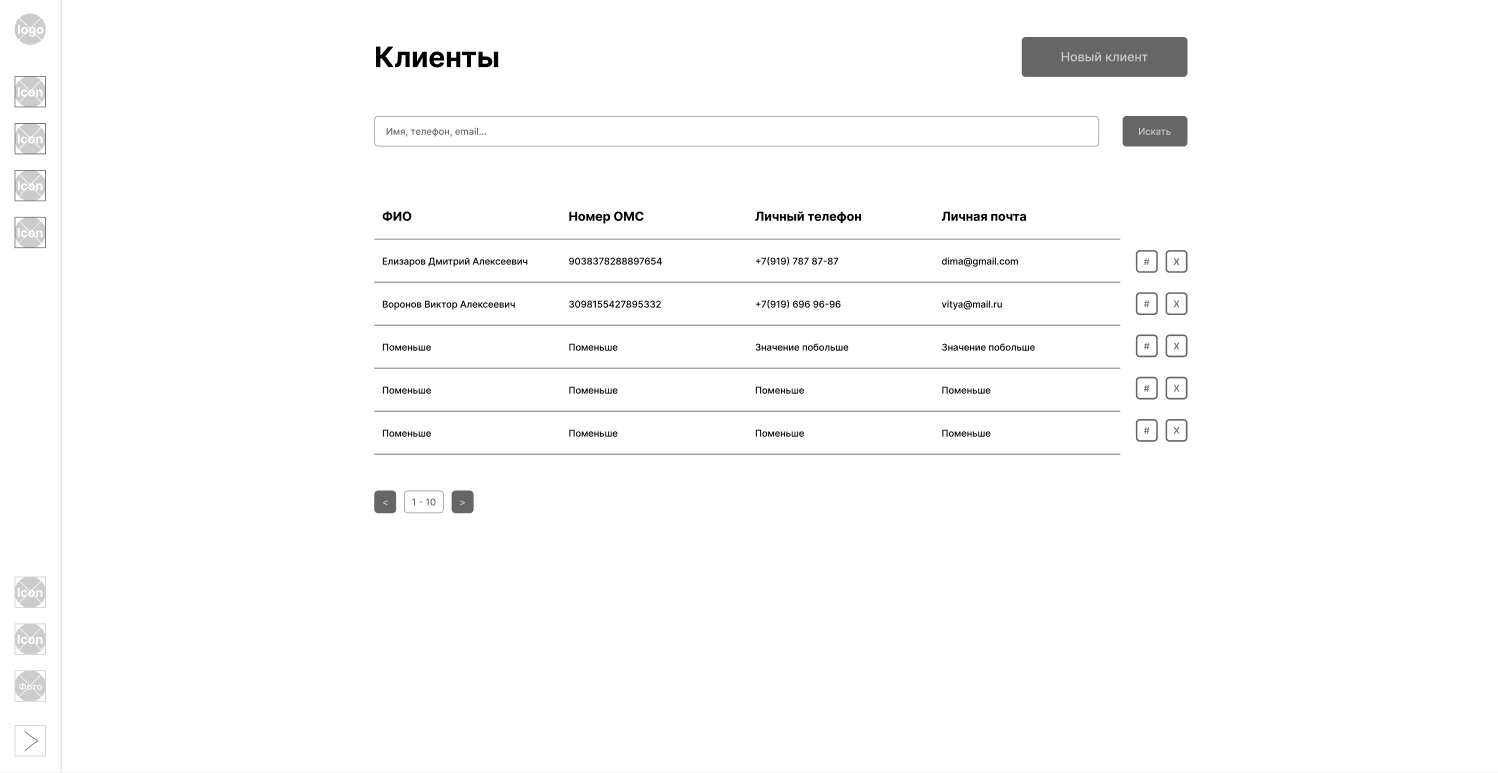
Данный интерфейс состоит из шапки профиля и основной части, реализует механизм изменения личных данных пользователя для авторизации в системе. Прототип интерфейса «Профиль пользователя» представлен на Рисунке 4.



***Рисунок 4 – Прототип страницы «Профиль пользователя»***

**2.7.3 Клиенты (пациенты)**

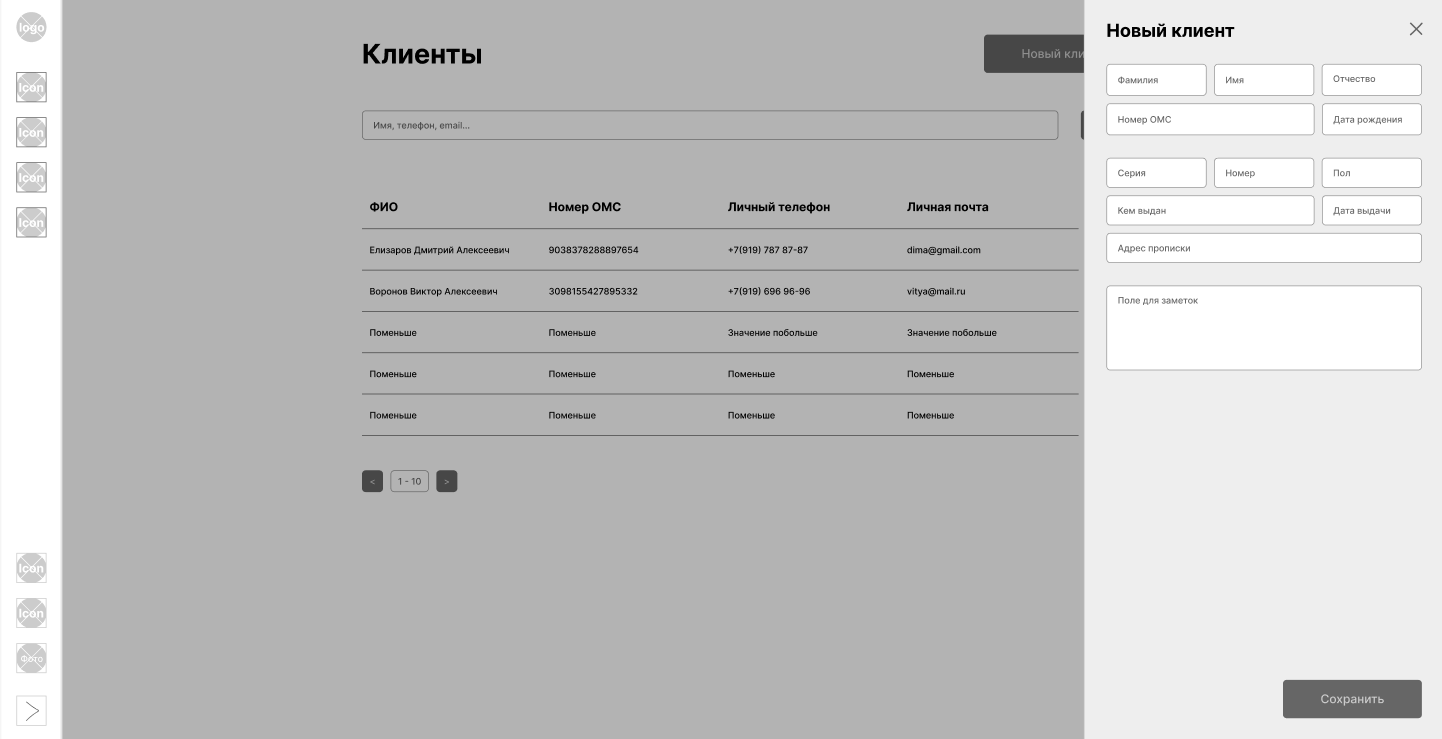
Данный интерфейс содержит таблицу со всеми клиентами текущего пользователя, поле поиска, пагинацию и кнопку добавления нового клиента. Реализует механизм просмотра карточки конкретного пациента. Прототип интерфейса «Клиенты» представлен на Рисунке 5.



***Рисунок 5 – Прототип страницы «Клиенты»***

**2.7.4 Добавление нового клиента**

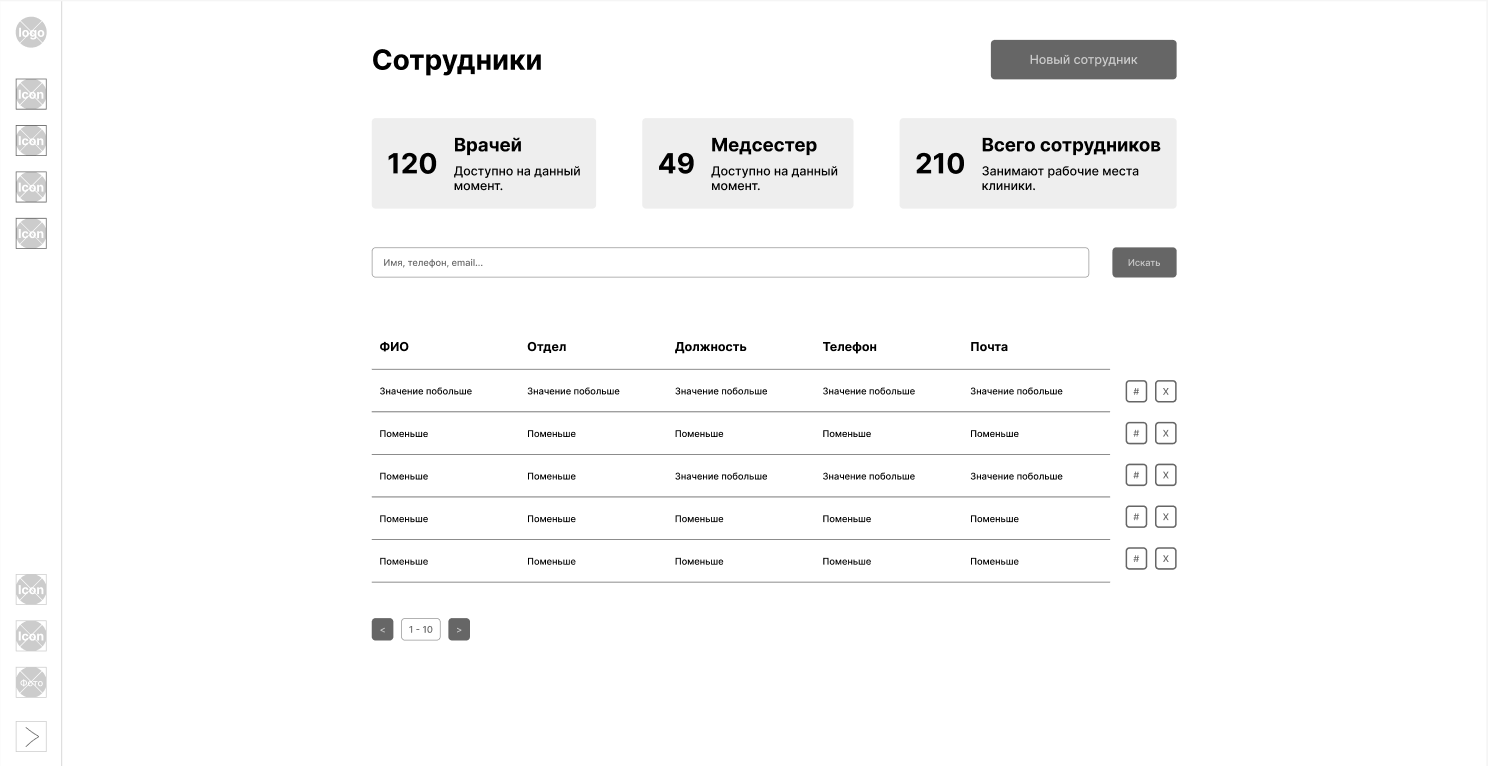
Данный интерфейс содержит поля для ввода основной информации об клиенте. Реализует механизм добавления нового клиента в систему. Прототип интерфейса «Добавление нового клиента» представлен на Рисунке 6.



***Рисунок 6 – Прототип страницы «Добавление нового клиента»***

**2.7.5 Сотрудники (пользователи)**

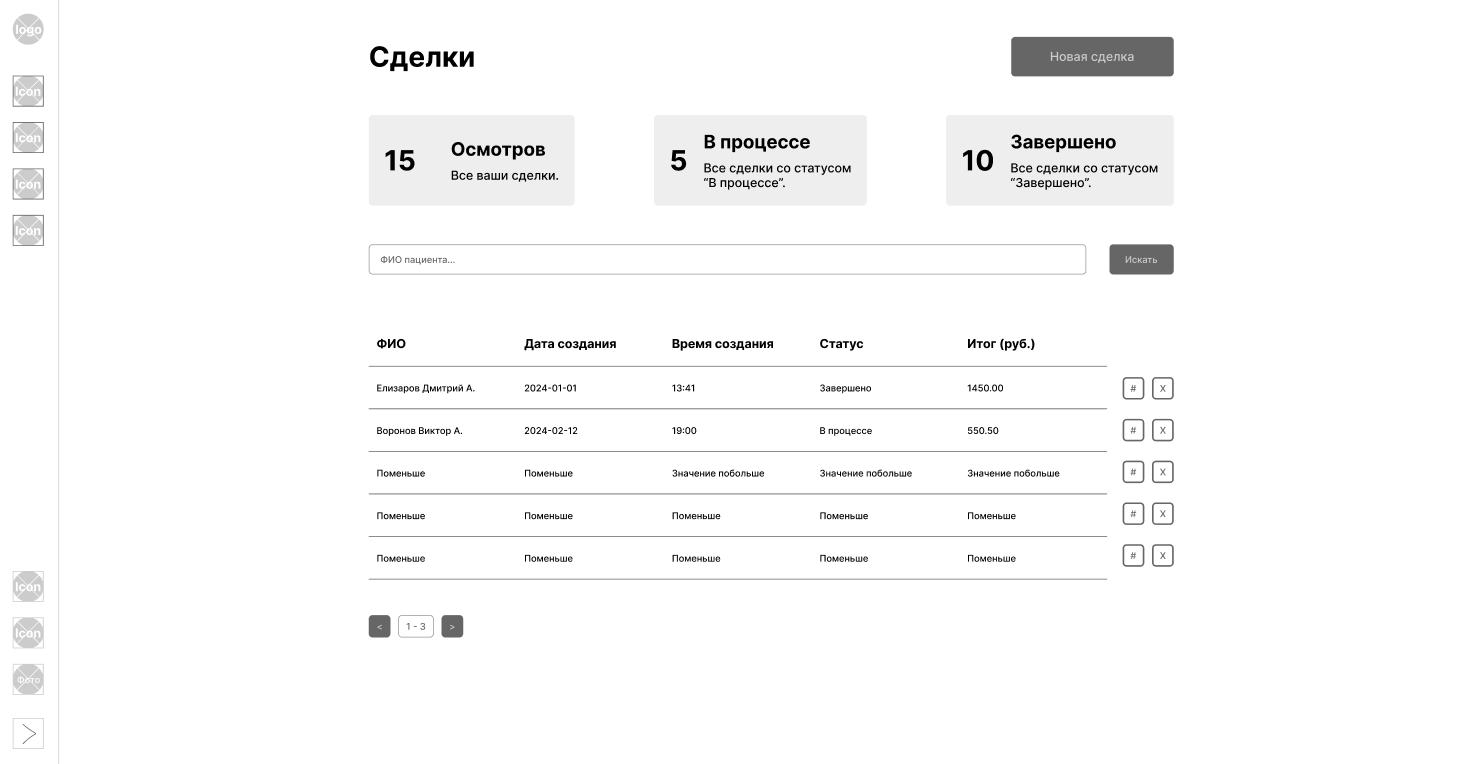
Данный интерфейс содержит таблицу со всеми существующими пользователями системы, поле поиска, пагинацию и кнопку добавления нового клиента. Реализует механизм просмотра карточки конкретного пользователя. Прототип интерфейса «Сотрудники» представлен на Рисунке 7.



***Рисунок 7 – Прототип страницы «Сотрудники»***

**2.7.6 Приёмы (осмотры)**

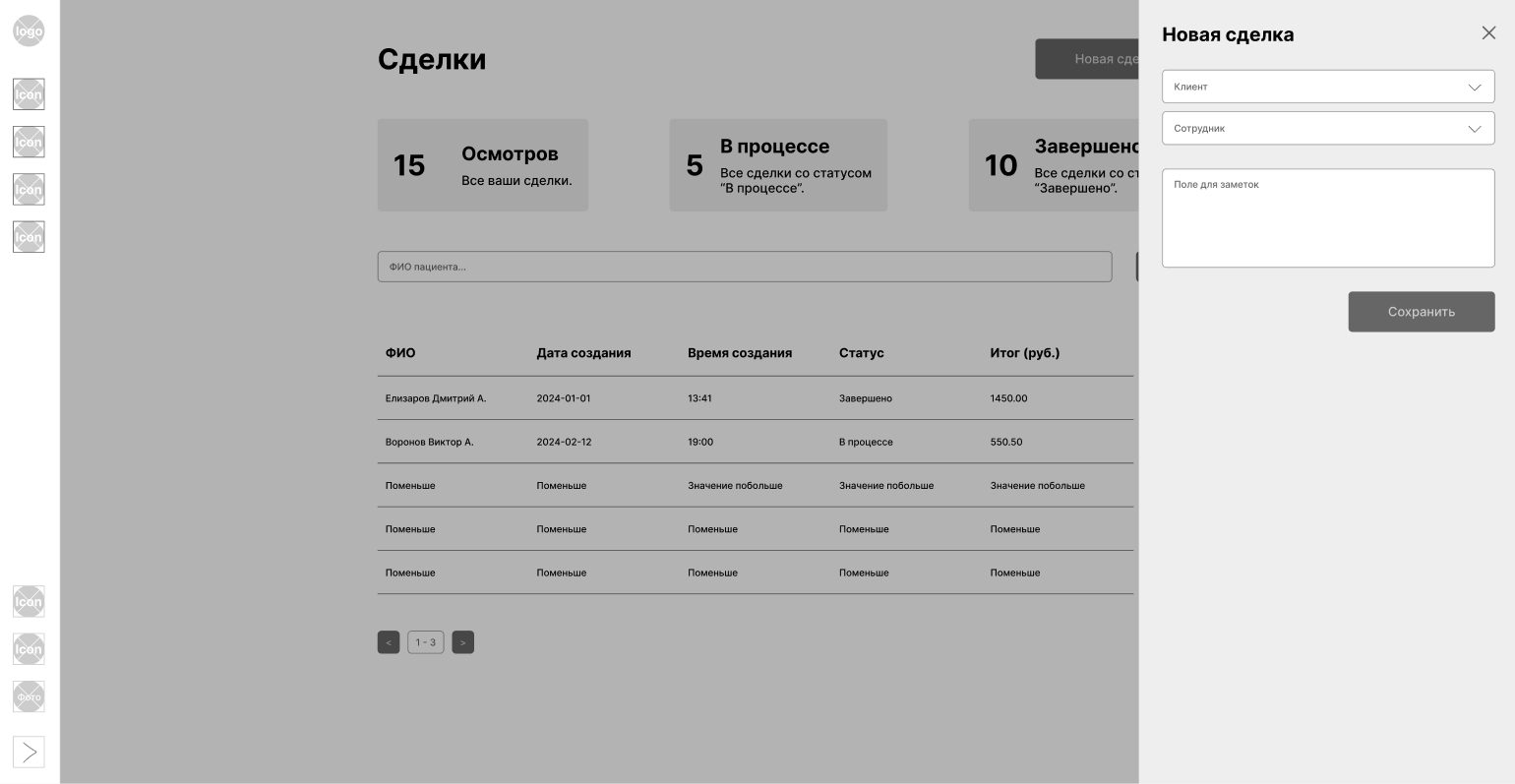
Данный интерфейс содержит таблицу со всеми существующими осмотрами конкретного пользователя системы, поле поиска, пагинацию и кнопку добавления нового осмотра. Реализует механизм просмотра карточки конкретного осмотра. Прототип интерфейса «Приёмы» представлен на Рисунке 8.



***Рисунок 8 – Прототип страницы «Приёмы»***

**2.7.7 Добавление нового приёма**

Данный интерфейс содержит поля для ввода основной информации о новом осмотре. Реализует механизм добавления нового осмотра в систему. Прототип интерфейса «Добавление нового осмотра» представлен на Рисунке 9.



***Рисунок 9 – Прототип страницы «Добавление нового приёма»***

## **Проектирование базы данных**

Проектирование базы данных является фундаментальным этапом в разработке информационных систем. Для корректной обработки имеющихся данных необходимо описать основные сущности и реализовать модель данных, описывающую концептуальные схемы выбранной предметной области. Для реализации данной модели будем использовать ER-модель (Entity-Relationship model).

Для начала необходимо провести приведение данных к трем основным формам представления: первая, вторая и третья начальные формы.

**2.8.1 Приведение к перовой нормальной форме**

Чтобы достичь первой нормальной формы, необходимо, чтобы данные представляли собой только простые скалярные значения, была соблюдена гарантированная уникальность значений, и в отношениях отсутствовали дублирующиеся строки. В результате этого процесса данные, соответствующие первой нормальной форме, будут выглядеть следующим образом (см. Таблицу 2).

Таблица 2 – Первая нормальная форма

| **Таблица** | **Поля** |
| --- | --- |
| Пользователи | Корпоративная почта |
| Телефон |
| Пароль |
| Идентификатор роли |
| Идентификатор отдела |
| Идентификатор должности |
| Идентификатор категории квалификации |
| Номер лицензии |
| Стаж работы |
| Фамилия |
| Имя |
| Отчество |
| Дата рождения |
| Пол |
| Серия/номер паспорта |
| Дата получения паспорта |
| Кем выдан паспорт |
| Адрес регистрации |
| Приёмы | Дата/время создания |
| Время закрытия |
| Цель визита |
| Диагноз |
| Назначения |
| Идентификатор пользователя |
| Идентификатор статуса осмотра |
| История болезни | Дата/время открытия |
| Дата/время закрытия |
| Идентификатор осмотра |
| Пациенты | Личная почта |
| Личный телефон |
| Идентификатор истории болезни |
| Номер ОМС (медицинское страхование) |
| Фамилия |
| Имя |
| Отчество |
| Дата рождения |
| Пол |
| Серия/номер паспорта |
| Дата получения паспорта |
| Кем выдан паспорт |
| Адрес регистрации |
| Роли | Наименование роли |
| Отделы | Наименование отдела |
| Должности | Наименование должности |
| Категория квалификации | Наименование категории |
| Статусы осмотров | Наименование статуса осмотра |

Таблицы «Роли», «Отделы», «Должности», «Категория квалификации» и «Статусы осмотров» вынесены в отдельные стержневые сущности.

**2.8.2 Приведение к второй нормальной форме**

Для того чтобы данные соответствовали второй нормальной форме, они должны быть уже в первой нормальной форме и также должно быть обеспечено следующее условие: каждый атрибут, не входящий в состав первичного ключа, должен зависеть только от полного первичного ключа, где первичный ключ представляет собой уникальный идентификатор. В результате этого процесса данные, соответствующие второй нормальной форме, будут выглядеть следующим образом (см. Таблицу 3).

Таблица 3 – Вторая нормальная форма

| **Таблица** | **Поля** |
| --- | --- |
| Пользователи | Идентификатор пользователя (первичный ключ) |
| Корпоративная почта |
| Телефон |
| Пароль |
| Идентификатор роли (внешний ключ) |
| Идентификатор отдела (внешний ключ) |
| Идентификатор должности (внешний ключ) |
| Идентификатор категории квалификации (внешний ключ) |
| Номер лицензии |
| Стаж работы |
| Фамилия |
| Имя |
| Отчество |
| Дата рождения |
| Пол |
| Серия/номер паспорта |
| Дата получения паспорта |
| Кем выдан паспорт |
| Адрес регистрации |
| Приёмы | Идентификатор осмотра (первичный ключ) |
| Дата/время создания |
| Время закрытия |
| Цель визита |
| Диагноз |
| Назначения |
| Идентификатор пользователя (внешний ключ) |
| Идентификатор статуса осмотра (внешний ключ) |
| История болезни | Идентификатор истории болезни (первичный ключ) |
| Дата/время открытия |
| Дата/время закрытия |
| Идентификатор осмотра (внешний ключ) |
| Пациенты | Идентификатор пациента (первичный ключ) |
| Личная почта |
| Личный телефон |
| Идентификатор истории болезни (внешний ключ) |
| Номер ОМС (медицинское страхование) |
| Фамилия |
| Имя |
| Отчество |
| Дата рождения |
| Пол |
| Серия/номер паспорта |
| Дата получения паспорта |
| Кем выдан паспорт |
| Адрес регистрации |
| Роли | Идентификатор роли (первичный ключ) |
| Наименование роли |
| Отделы | Идентификатор отдела (первичный ключ) |
| Наименование отдела |
| Должности | Идентификатор должности (первичный ключ) |
| Наименование должности |
| Категория квалификации | Идентификатор категории квалификации (первичный ключ) |
| Наименование категории |
| Статусы осмотров | Идентификатор статуса осмотра (первичный ключ) |
| Наименование статуса осмотра |

Были добавлены индивидуальные уникальные идентификаторы (первичные ключ) и связывающие отношения между собой идентификаторы-ссылки (внешние ключи).

**2.8.3 Приведение к третьей нормальной форме**

Для того чтобы данные соответствовали третьей нормальной форме, они должны соответствовать второй нормальной форме, и каждый атрибут должен быть функционально зависим только от первичного ключа. В результате этого процесса данные, соответствующие третьей нормальной форме, будут выглядеть следующим образом (см. Таблицу 4).

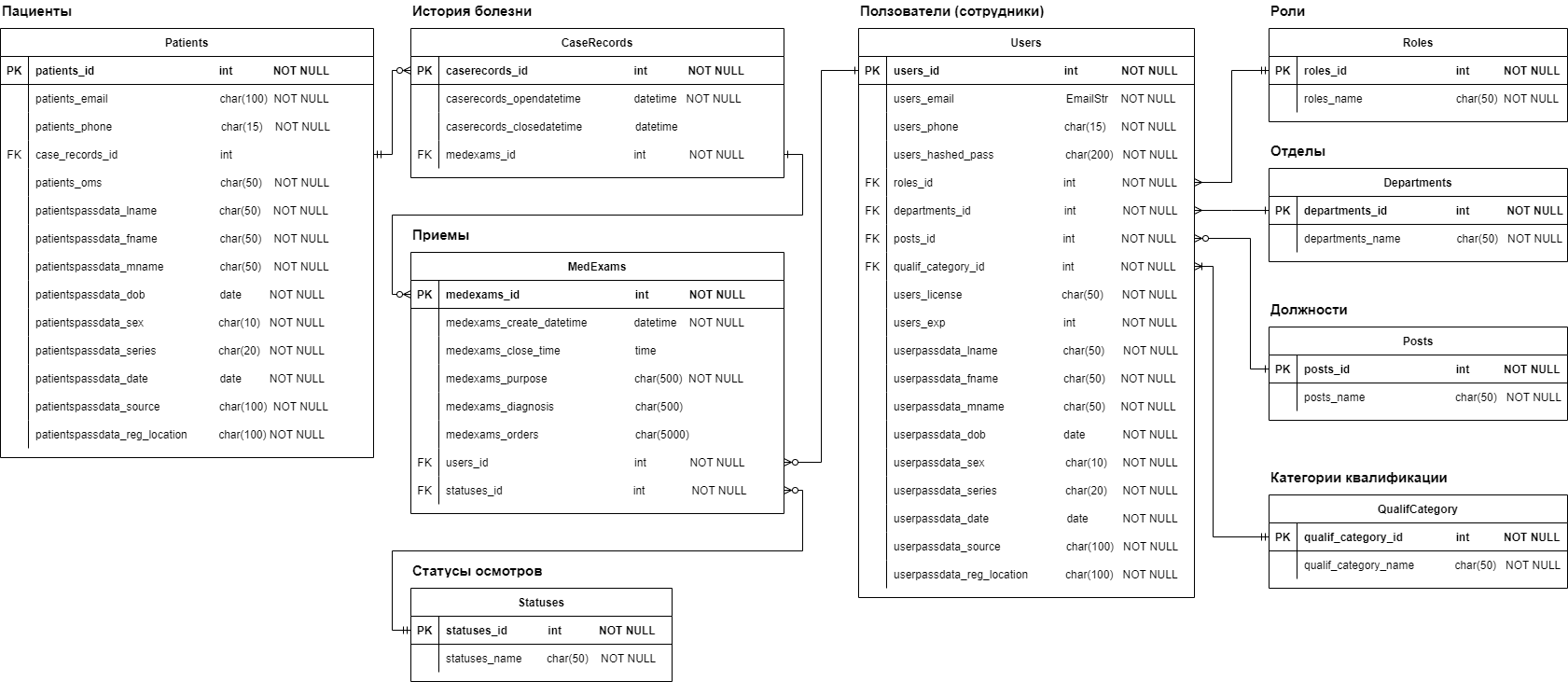
Таблица 4 – Третья нормальная форма

| **Таблица** | **Поля** |
| --- | --- |
| Users | Users\_id <pk> |
| Users\_email |
| Users\_phone |
| Users\_hashed\_pass |
| Roles\_id <fk> |
| Departments\_id <fk> |
| Posts\_id <fk> |
| Qualif\_category\_id <fk> |
| Users\_license |
| Users\_exp |
| Users\_lname |
| Users\_fname |
| Users\_mname |
| Users\_dob |
| Users\_sex |
| Users\_series |
| Users\_date |
| Users\_source |
| Users\_reg\_location |
| MedExams | MedExams\_id <pk> |
| MedExams\_create\_datetime |
| MedExams\_close\_time |
| MedExams\_purpose |
| MedExams\_diagnosis |
| MedExams\_orders |
| Users\_id <fk> |
| Statuses\_id <fk> |
| CaseRecords | CaseRecords\_id <pk> |
| CaseRecords\_open\_datetime |
| CaseRecords\_close\_datetime |
| MedExams\_id <fk> |
| Patients | Patients\_id <pk> |
| Patients\_email |
| Patients\_phone |
| CaseRecords\_id <fk> |
| Patients\_oms |
| Patients\_lname |
| Patients\_fname |
| Patients\_mname |
| Patients\_dob |
| Patients\_sex |
| Patients\_series |
| Patients\_date |
| Patients\_source |
| Patients\_reg\_location |
| Roles | Roles\_id <pk> |
| Roles\_name |
| Departments | Departments\_id <pk> |
| Departments\_name |
| Posts | Posts\_id <pk> |
| Posts\_name |
| QualifCategory | QualifCategory\_id <pk> |
| QualifCategory\_name |
| Statuses | Statuses\_id <pk> |
| Statuses\_name |

Для точного определения связи конкретного поля и конкретной таблицы было принято решение давать специальные наименования полям в формате «Отношение\_поле». Данный способ обозначения полей позволит упросить разработку серверной части веб-приложения.

**2.8.4 ER-диаграмма базы данных**

В результате процесса приведения базы данных ко всем трем нормальным формам и процесса нормализации, получена декомпозированная база данных: определены ключи, и для каждого ключа установлены связи между соответствующими таблицами (см. Рисунок 10).



***Рисунок 10 – ER-диаграмма***

# **Глава 3. Реализация**

Веб-приложение к базе данных было разработано с использование ряда современных технологических решений. Серверная часть информационной системы была реализована при помощи недавно появившегося фреймворка FastAPI (Python), клиентская часть была реализована при помощи веб-библиотеки ReactJS (TypeScript).

Для реализации базы данных было принято решение использовать наиболее популярную СУБД – PostgreSQL. Тестирование базы данных проводилось через графический интерфейс pgAdmin 4.

## **Выбор технологий разработки**

При разработке медицинской информационной системы были выбраны современные технологические решения. Важными факторами выбора тех или иных решений были: доступность, легкость в первом вхождении, популярность и наличие качественной и подробной документации.

**3.1.1 Серверная часть веб приложения**

Для разработки серверной части информационной системы был выбран фреймворк на языке Python под названием «FastAPI», который набрал большую популярность за последние несколько лет.

API (application programming interface) представляет собой набор доступных функций приложения, которые обеспечивают возможность взаимодействия с ним. В контексте сложных и объемных приложений API служит удобным и понятным для пользователя средством для работы с функционалом приложения.

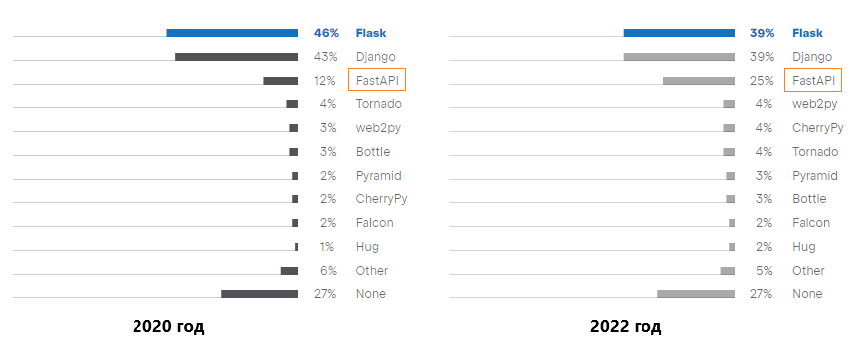
FastAPI - это фреймворк на языке Python, который обеспечивает быстроту и краткость написания эндпоинтов (конечные точки серверной части, к которым обращается клиентская часть для работы с данными из базы) [4]. Данный фреймворк был разработан в 2018 году одним независимым разработчиком, поддерживается и развивается до сих пор.

Определим основные преимущества и недостатки данного решения (см. Таблицу 5).

Таблица 5 – Преимущества и недостатки FastAPI

|  |  |
| --- | --- |
| **Преимущества** | **Недостатки** |
| Подробная документация на русском языке | Небольшое количество обучающих материалов в сети Интернет |
| Простота освоения |
| Встроенная валидация данных |
| Асинхронность | Небольшое количество дополнительных расширений и библиотек |
| Автогенерируемая документация (через Swagger UI) |
| Рост популярности за последние несколько лет |

Стоит уделить особое внимание росту популярности данного фреймворка. Если обратиться к результатам опроса от компании JetBrains «Python Developer Survey» 2020 и 2022 годов, то можно заметить резкий рост популярности данного решения (см. Рисунок 11) [5, 6].



***Рисунок 11 – опрос «Python Developer Survey» за 2020 и 2022 год***

Можно сделать вывод, что за 2 года популярность фреймворка FastAPI среди Python-разработчиков возросла на 13%. С каждым годом данное программное решение развивается и набирает популярность, что в свою очередь является хорошим поводом для углубленного изучения данной технологии.

**3.1.2 Клиентская часть веб приложения**

Для разработки клиентской части информационной системы была выбран веб-библиотека «ReactJS». Язык реализации системы – «TypeScript» (типизированная версия JavaScript).

Выбор данной веб-технологии обусловлен высокой популярностью и большим количеством обучающего материала. Определим основные преимущества и недостатки данного решения (см. Таблицу 6).

Таблица 6 – Преимущества и недостатки ReactJS

|  |  |
| --- | --- |
| **Преимущества** | **Недостатки** |
| Виртуальное DOM дерево | Сложность освоения |
| Компонентный подход | Отсутствие официального набора инструментов и правил |
| Широкая поддержка | Необходимость в использовании дополнительных инструментов |
| Однонаправленный поток данных |
| Высокая популярность среди веб-разработчиков |

При разработке клиентской части информационной системы использовалось средство разработки веб-приложений – TypeScript. Данный язык был выбран в связи с наличием строгой типизацией и схожим с синтаксисом языка JavaScript.

**3.1.3 Среда разработки**

В процессе разработки медицинской информационной системы в качестве основной среды использовалась «Visual Studio Code» (VSCode). Данное решение было принято в связи с широким инструментарием разработки и обширной настройкой данной интегрированной среды.

Особо стоит выделить возможность добавления и глубокой настройки различных расширений среды, которые значительно упрощают и ускоряют разработку. Перечень использованных дополнений представлен в Таблице 7.

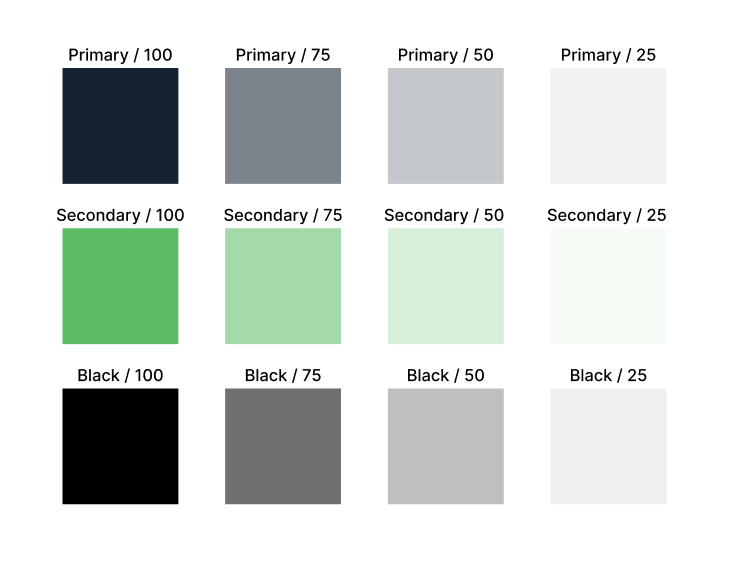
Таблица 7 – Расширения для Visual Studio Code

|  |  |
| --- | --- |
| **Наименование** | **Функционал** |
| Vim | Эмулятор редактора Vim (ускоряет слепую печать) |
| Tailwind CSS IntelliSense | Автодополнение, подцветка синтаксиса, функционал линтера для TailwindCSS (отображение ошибок) |
| Prettier | Форматирование TypeScript кода |
| PostgreSQL | Работа с БД на базе PostgreSQL через среду разработки |
| Docker | Работа с контейнеризацией (необходимо для развертывания) |
| Auto Rename Tag | Автоматическое закрытие тегов для языков HTML и XML |
| Black Formatter | Форматирование кода Python по стандартам PEP-8 |

Данная среда разработки позволила быстро и комфортно разработать полноценный программный продукт, протестировать и развернуть информационную систему.

## **Разработка дизайн системы**

Перед началом создания клиентской части информационной системы, необходимо разработать дизайн-систему. В дальнейшем это поможет правильно определять цветовые сочетания и не тратить на это время. На Рисунке 12 представлены палитры цветов для верстки системы. Цвета разделены на градации по степени прозрачности: 100%, 75%, 50%, 25%.



***Рисунок 12 – дизайн система***

Имеются градации основного (primary), вторичного (secondary) и черного цветов.

Для завершения дизайн системы необходимо создать логотип компании. При создании логотипа был использован бесплатный сервис logo.com, который предоставляет услуги генерации разнообразных логотипов и шрифтов по названию компании [7]. На рисунке 13 изображен полученный генеративным образом логотип.



***Рисунок 13 – логотип компании разработчика***

Шрифт для всей системы был выбран из открытой библиотеки шрифтов Google Fonts – Rubik [8]. Простой и практичный шрифт с различными начертаниями: light, regular, medium, semibold, bold.

## **Разработка серверной части**

Основной задачей разработки серверной части информационной системы является определение основных запросов к базе данных. Для этого необходимо сначала создать базу данных и заполнить её тестовыми данными.

**3.3.1 Создание моделей данных**

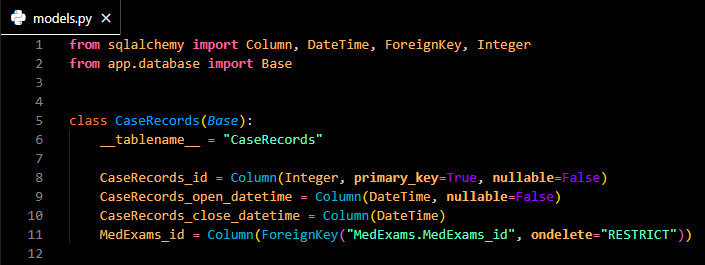
Для описания моделей сущностей использовалась библиотека под названием «SQLAlchemy». Данная библиотека позволяет описать модели отношений, связи между ними и необходимые ограничения через язык программирования (в данном случае через Python).

Затем необходимо прокинуть миграции на сервер выбранной базы данных (в данном случае PostgreSQL) для создания всех спроектированных таблиц. Для этого была использована библиотека для Python под названием «Alembic». Данная библиотека позволяет связать описанные через SQLAlchemy модели и реальную БД.

Данная связка библиотек позволяет легко создавать и поддерживать БД напрямую через среду разработки, что в свою очередь уменьшает время разработки.

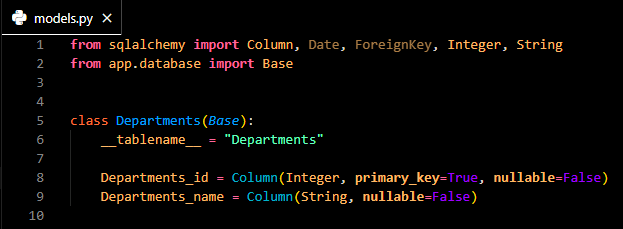
Опишем все существующие отношение БД в виде моделей SQLAlchemy.

На Рисунке 14 представлена модель истории болезни – CaseRecords.



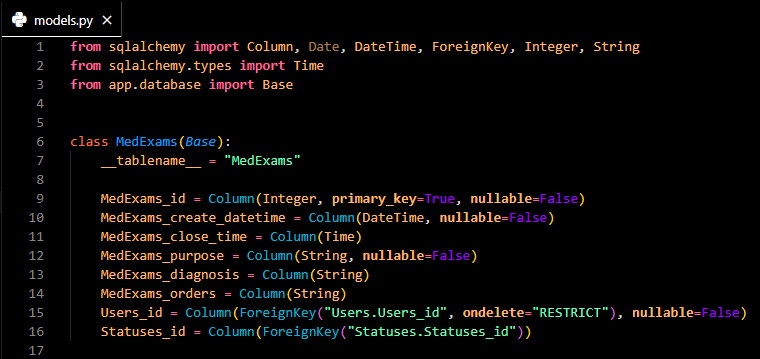
***Рисунок 14 – модель «CaseRecords»***

На Рисунке 15 представлена модель отделов – Departments.



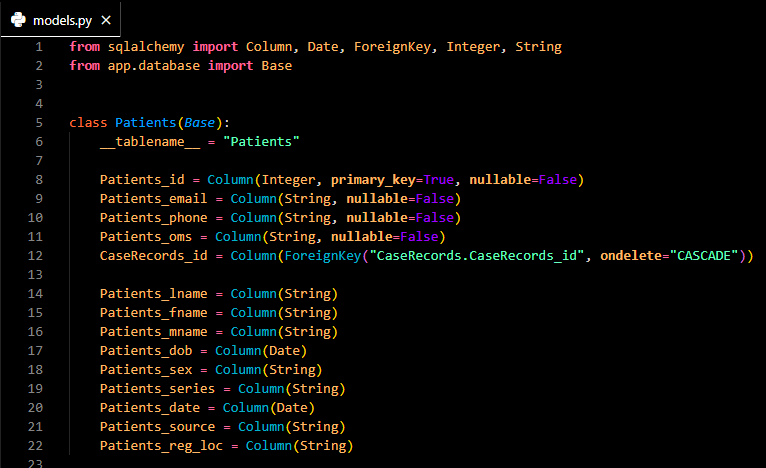
***Рисунок 15 – модель «Departments»***

На Рисунке 16 представлена модель осмотров – MedExams.



***Рисунок 16 – модель «MedExams»***

На Рисунке 17 представлена модель клиентов (пациентов) – Patients.



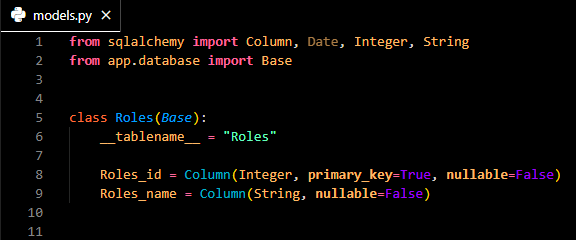
***Рисунок 17 – модель «Patients»***

На Рисунке 18 представлена модель должностей – Posts.



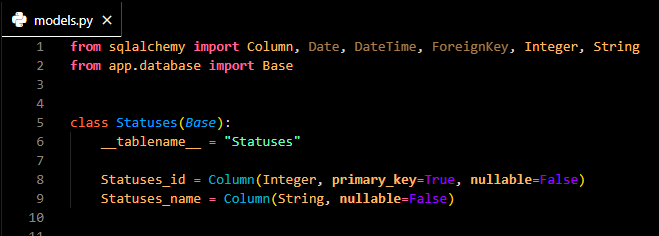
***Рисунок 18 – модель «Posts»***

На Рисунке 19 представлена модель ролей – Roles.



***Рисунок 19 – модель «Roles»***

На Рисунке 20 представлена модель статусов осмотров – Statuses.



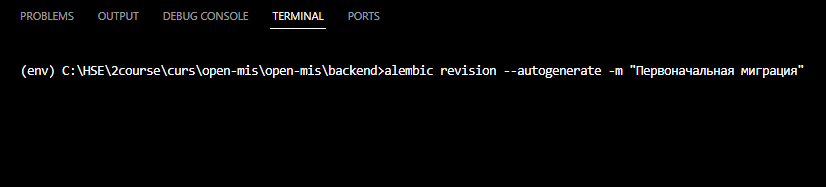
***Рисунок 20 – модель «Statuses»***

На Рисунке 21 представлена модель пользователей – Users.



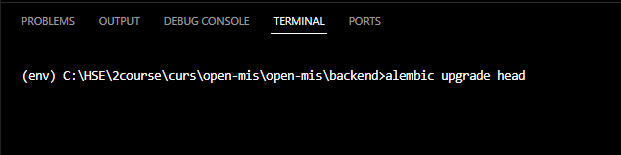
***Рисунок 21 – модель «Users»***

После описания всех моделей сущностей необходимо прокинуть миграцию и создать таблицы внутри базы данных. Для этого воспользуемся библиотекой Alembic и введем следующие 2 команды (см. Рисунок 22 и Рисунок 23).



***Рисунок 22 – создание миграции***

Данная команда создаст миграцию для дальнейшего создания отношений на её описании.



***Рисунок 23 – прокидывание миграции***

Данная команда отправит только что созданную миграцию на сервер базы данных PostgreSQL и создаст все описанные в виде моделей SQLAlchemy таблицы.

**3.3.2 Конечные узлы системы и запросы к БД**

После создания базы данных и заполнения тестовых данных необходимо прописать конечные узлы (endpoints) для связи с клиентской частью. Для этого воспользуемся надстройкой (ORM) над языком SQL – «SQLAlchemy», которую мы использовали до этого в качестве инструмента описания моделей сущностей.

Опишем все существующие конечные узлы разрабатываемой медицинской информационной системы.

Узлы «CaseRecords» обрабатывают следующие запросы:

* получение истории болезни по идентификатору пользователя;
* закрытие истории болезни;
* связка конкретной истории болезни и конкретного осмотра.

Узлы «MedExams» обрабатывают следующие запросы:

* получение информации про все осмотры конкретного пользователя;
* получение карточки конкретного осмотра;
* добавление нового осмотра;
* редактирование информации конкретного осмотра;
* удаление конкретного осмотра.

Узлы «Patients» обрабатывают следующие запросы:

* получение информации про всех клиентов конкретного пользователя;
* получение карточки конкретного клиента;
* добавление нового клиента;
* редактирование информации конкретного клиента;
* удаление конкретного клиента.

Узлы «profile» обрабатывают следующие запросы:

* получение основной информации для шапки профиля;
* изменение пароля для входа в систему.

Узлы «Roles» обрабатывают следующие запросы:

* получение всех ролей системы;
* получение всех отделов системы;
* получение всех должностей системы;
* получение всех статусов осмотров.

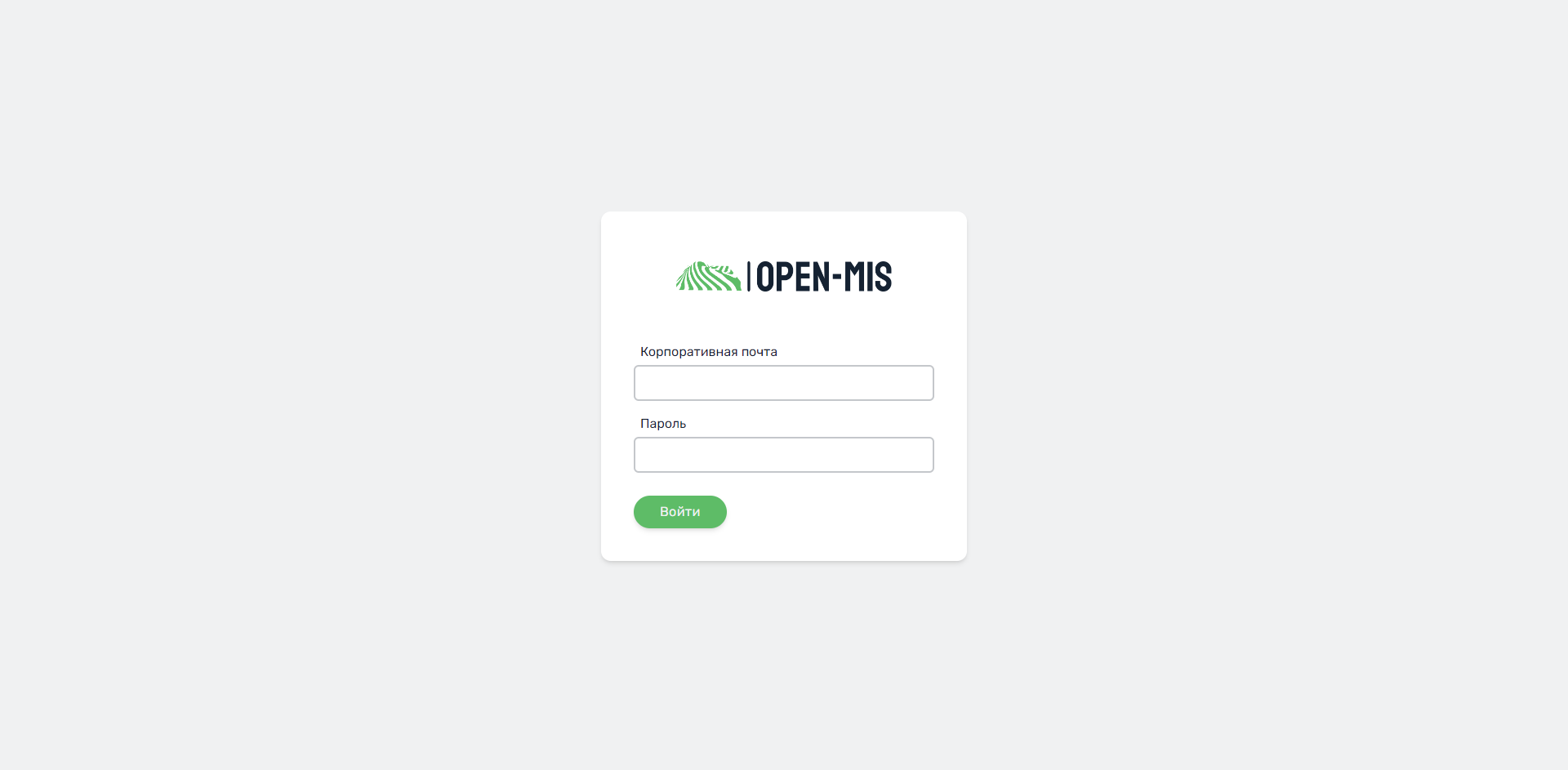
Узлы «Users» обрабатывают следующие запросы:

* получение идентификационного номера текущего пользователя;
* вход в систему;
* выход из системы;
* получение информации про всех пользователей системы;
* получение карточки конкретного пользователя;
* добавление нового пользователя;
* редактирование информации конкретного пользователя;
* удаление конкретного пользователя.

## **Разработка клиентской части**

Основной задачей разработки клиентской части информационной системы является верстка интерфейсов и привязка серверной части. Далее представлены результаты верстки страниц системы на языке TSX (препроцессор HTML) и TailwindCSS (препроцессор CSS).

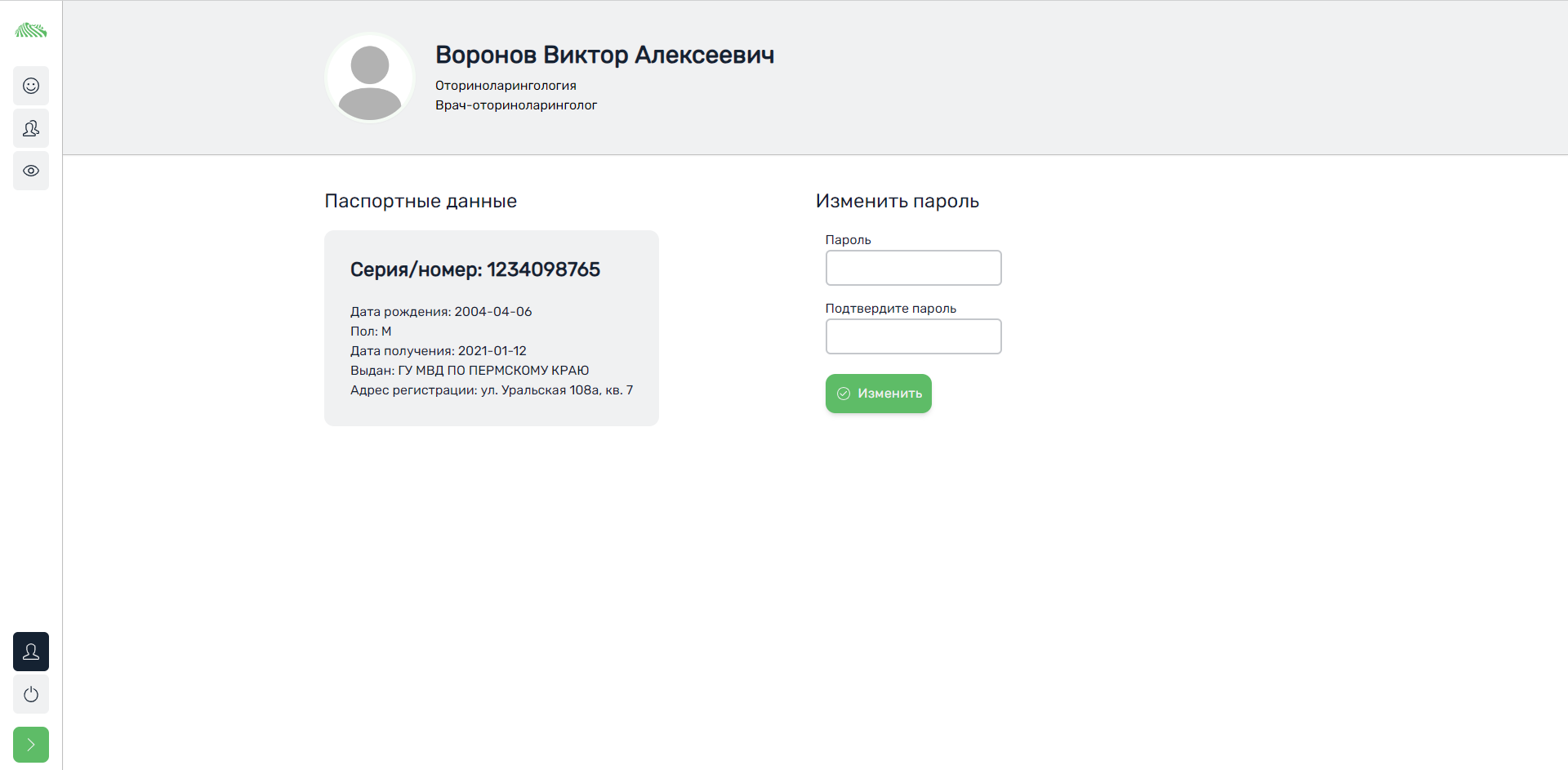
На Рисунке 24 изображена страница входа в систему.



***Рисунок 24 – верстка страницы входа в систему***

На данную страницу добавлен логотип компании, сгенерированный при помощи бесплатного сервиса logo.com, который предоставляет услуги генерации разнообразных логотипов и цветовых решений по названию компании [7].

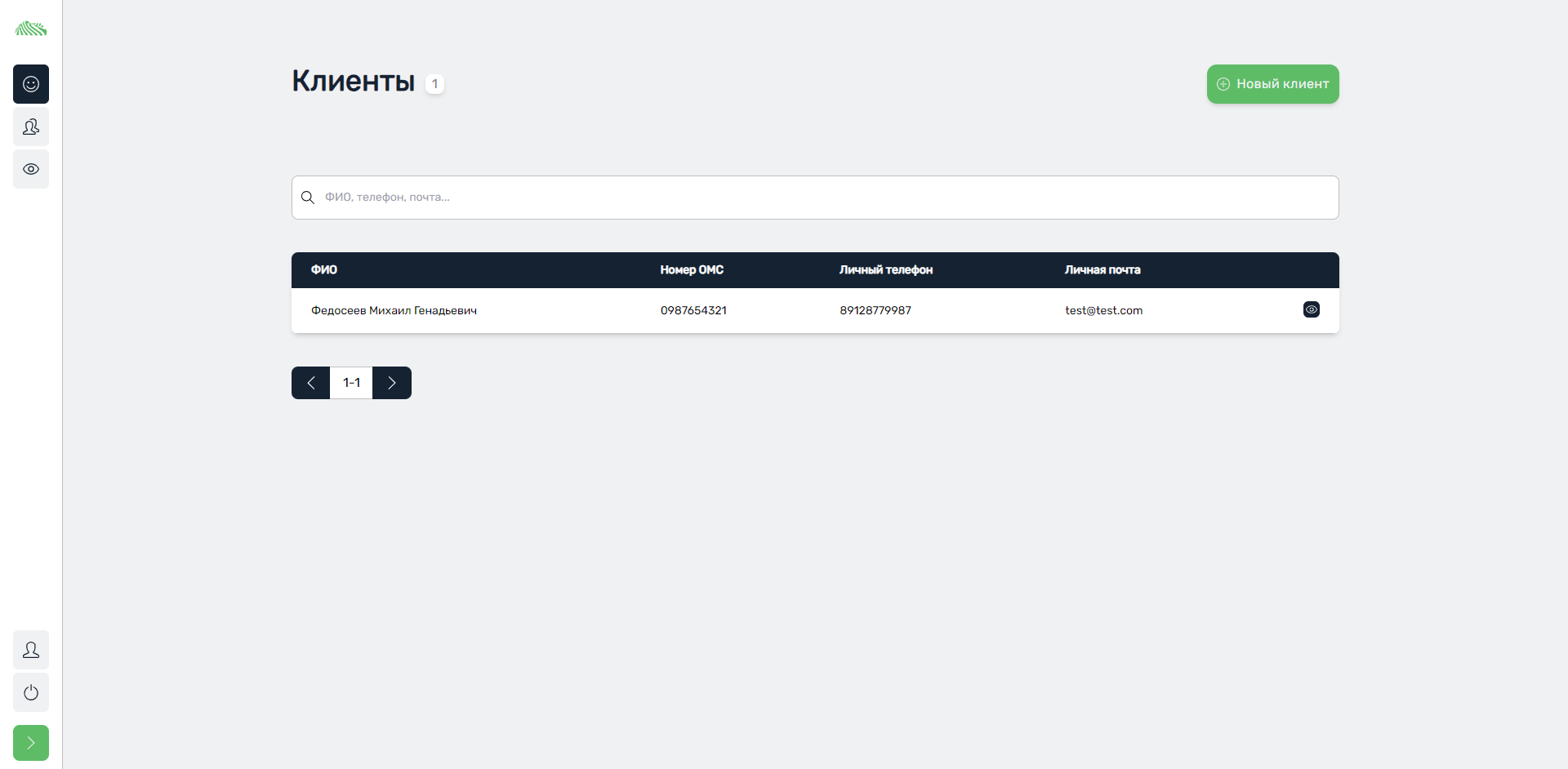
На Рисунке 25 изображена страница профиля пользователя.



***Рисунок 25 – верстка страницы профиля пользователя***

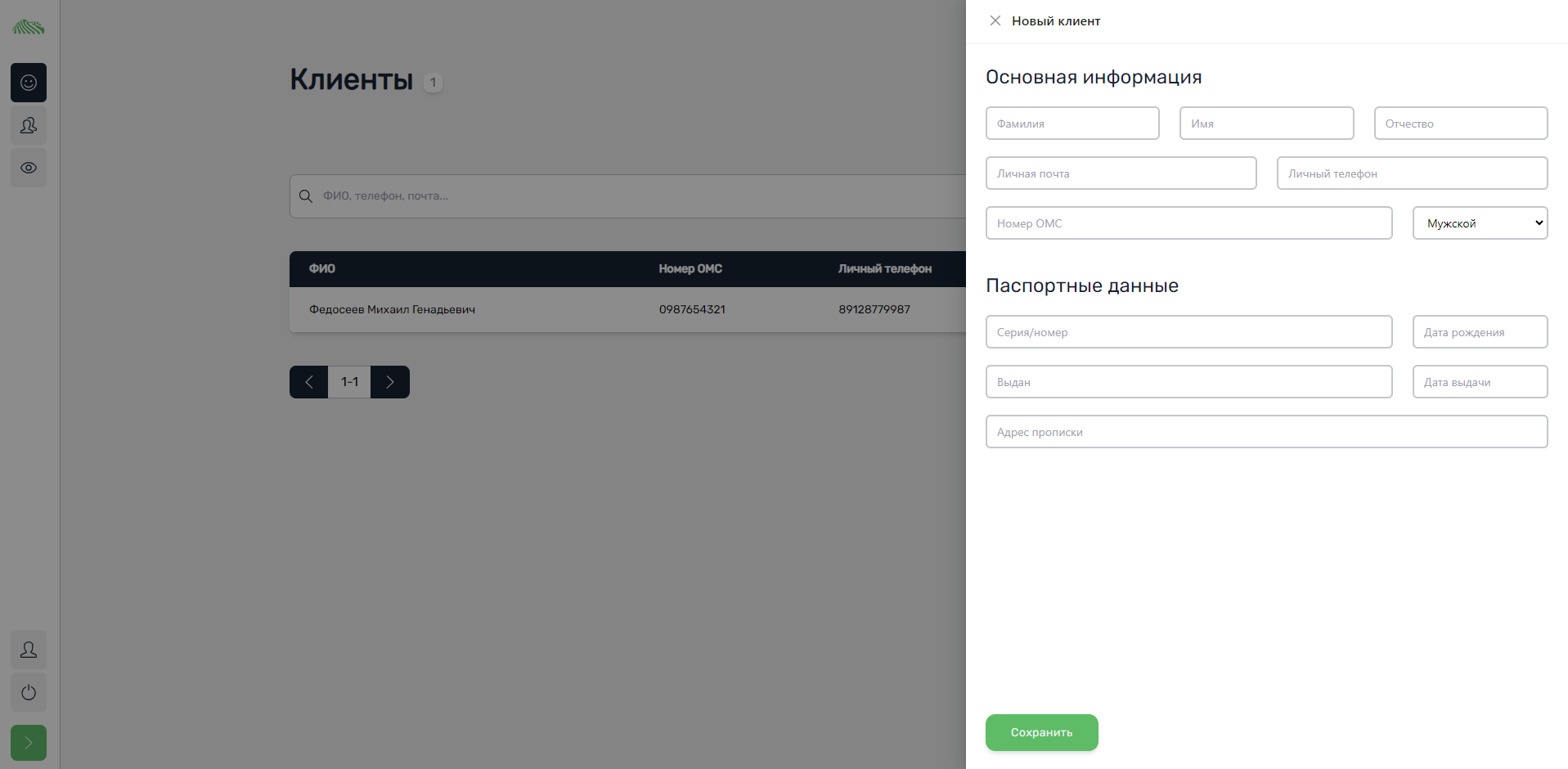
На данной странице изменен подход к обновлению пользовательских данных для входа. Пользователь может изменить только пароль, в то время как корпоративная почта является неизменяемой.

На Рисунке 26 изображена страница клиентов.



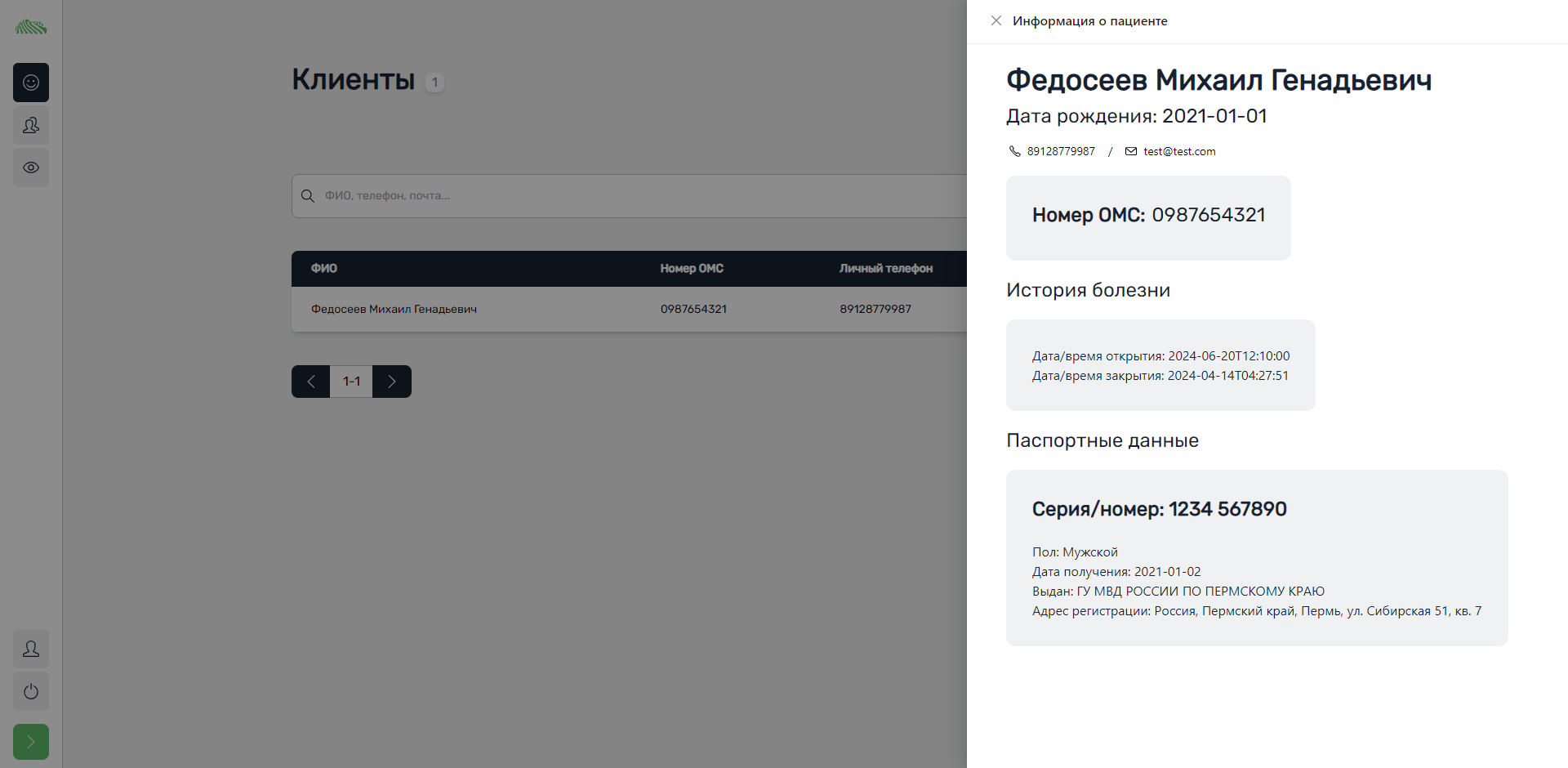
***Рисунок 26 – верстка страницы клиентов***

На Рисунке 27 изображена форма добавление нового клиента.



***Рисунок 27 – верстка формы добавления нового клиента***

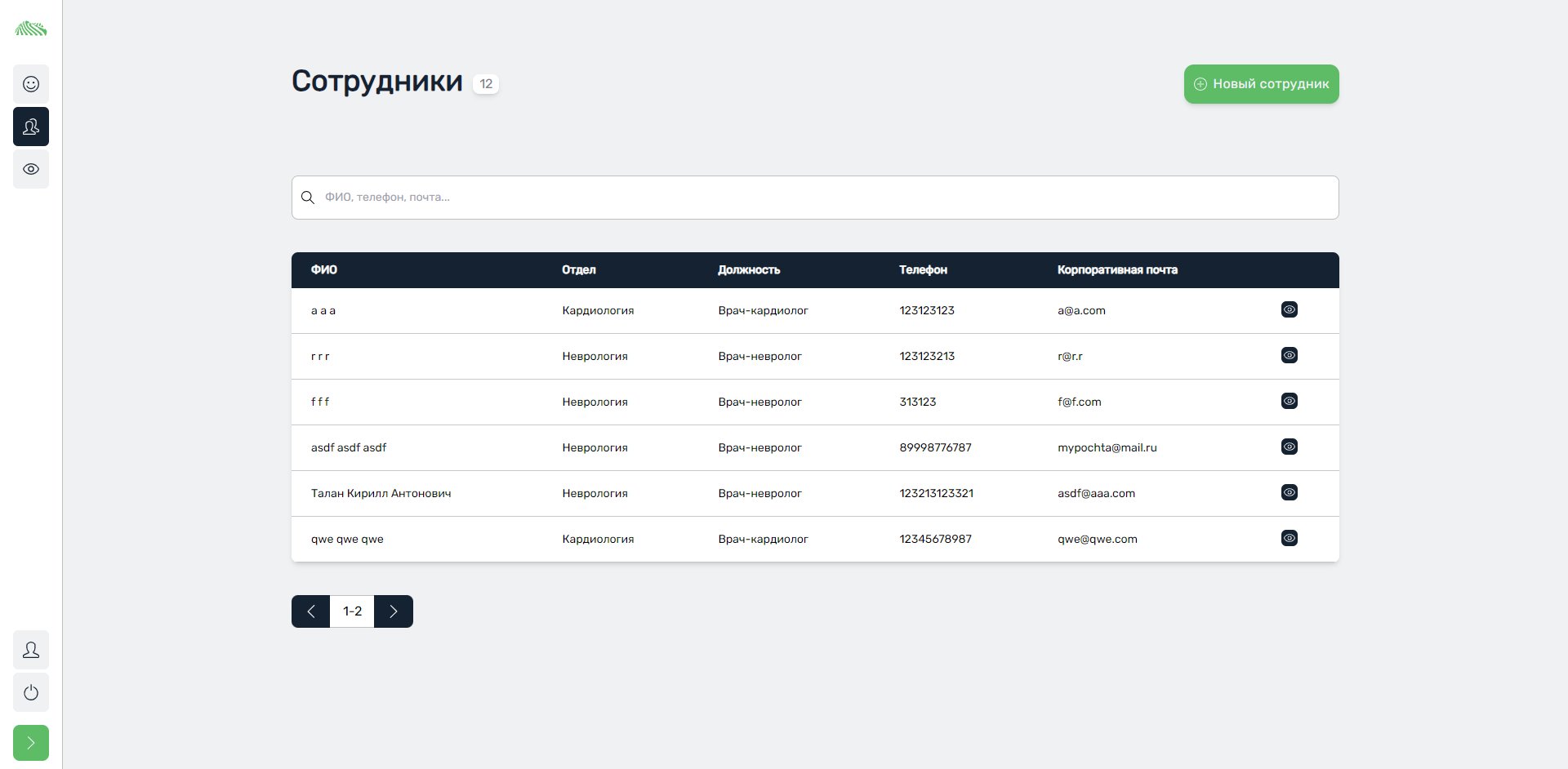
На Рисунке 28 изображен элемент карточки конкретного клиента.



***Рисунок 28 – верстка элемента карточки клиента***

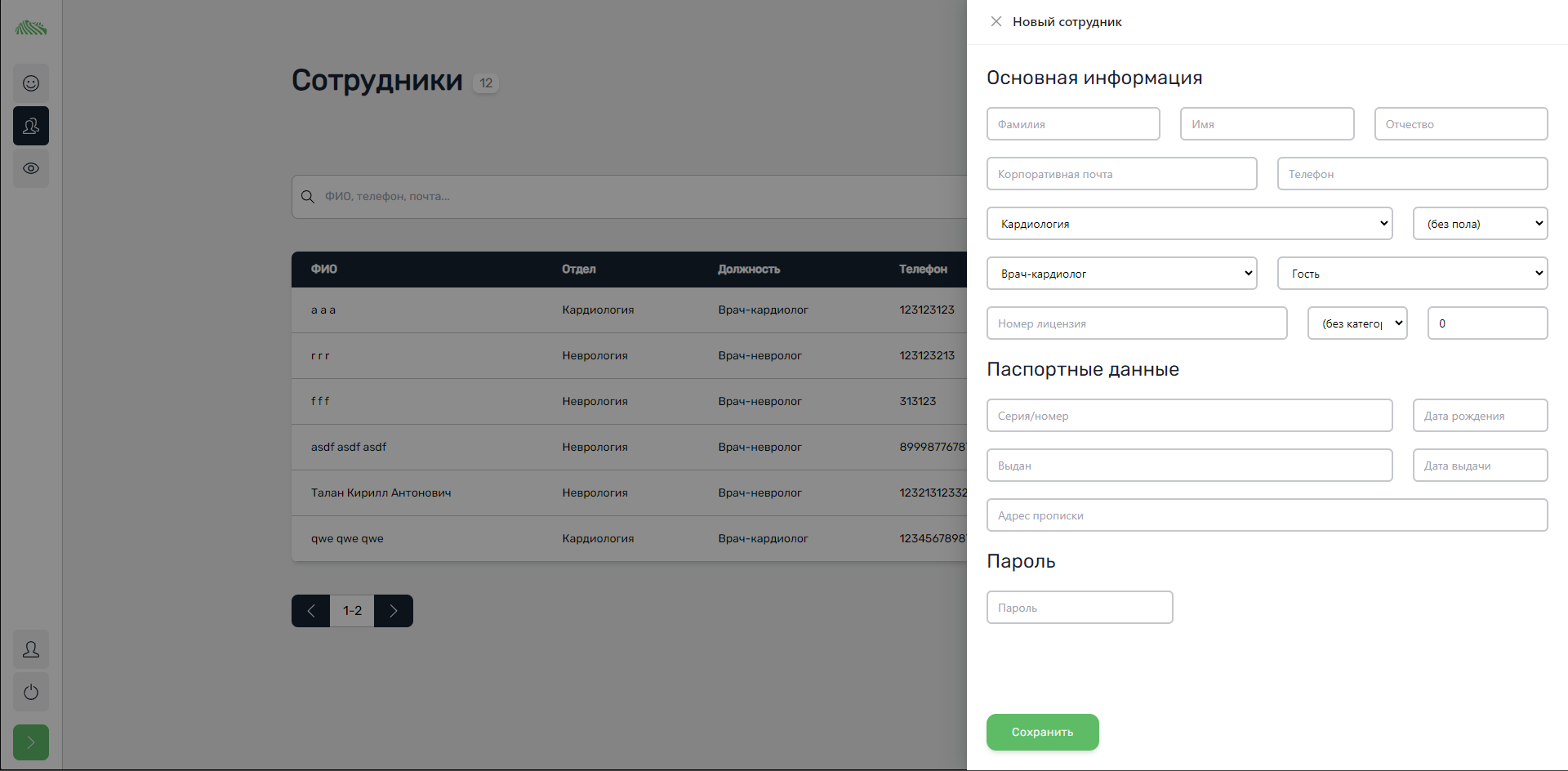
Данный элемент предоставляет все необходимые данные о конкретном клиенте.

На Рисунке 29 изображена страница сотрудников.



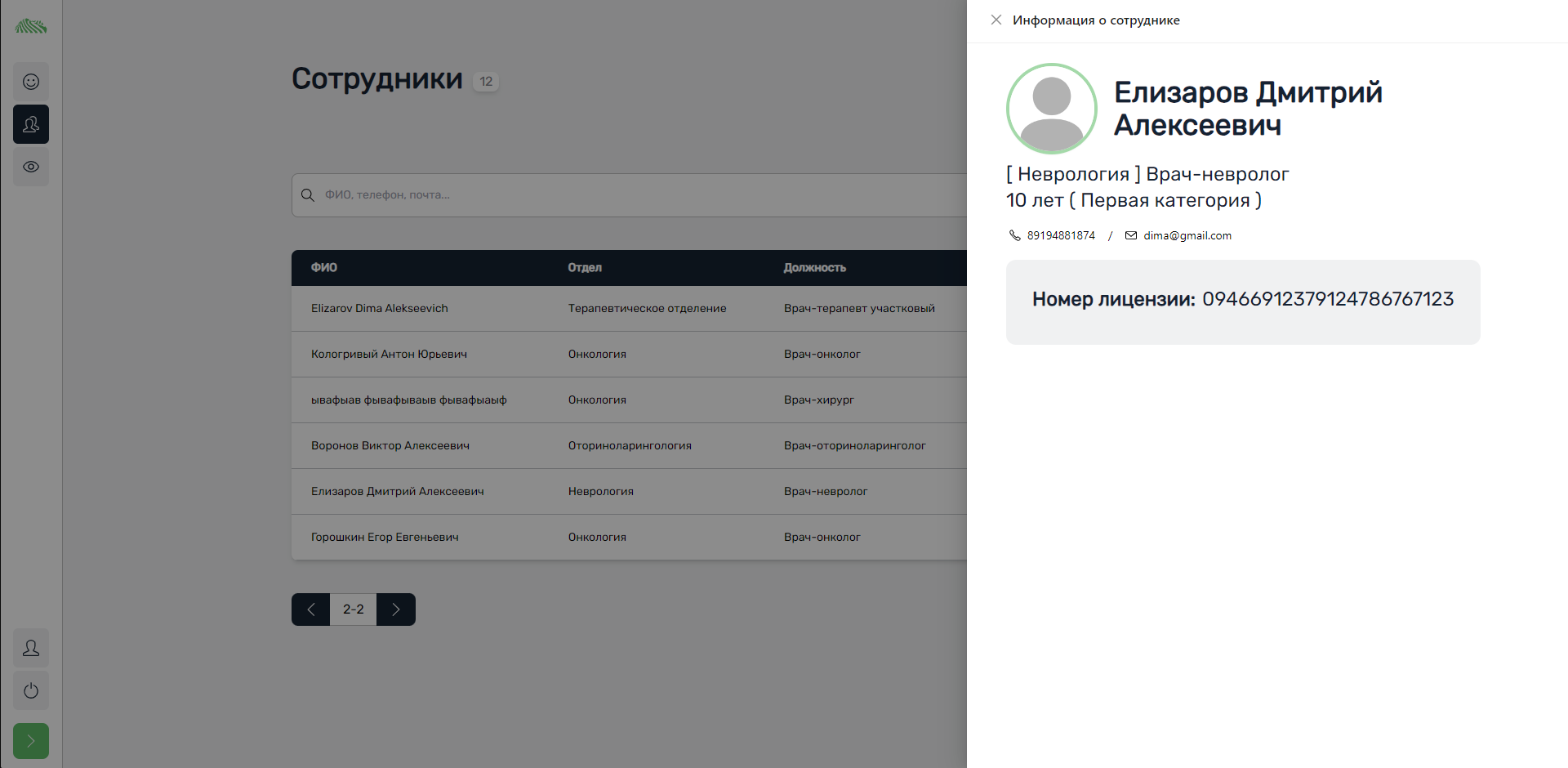
***Рисунок 29 – верстка страницы сотрудников***

На Рисунке 30 изображена форма создания нового сотрудника.



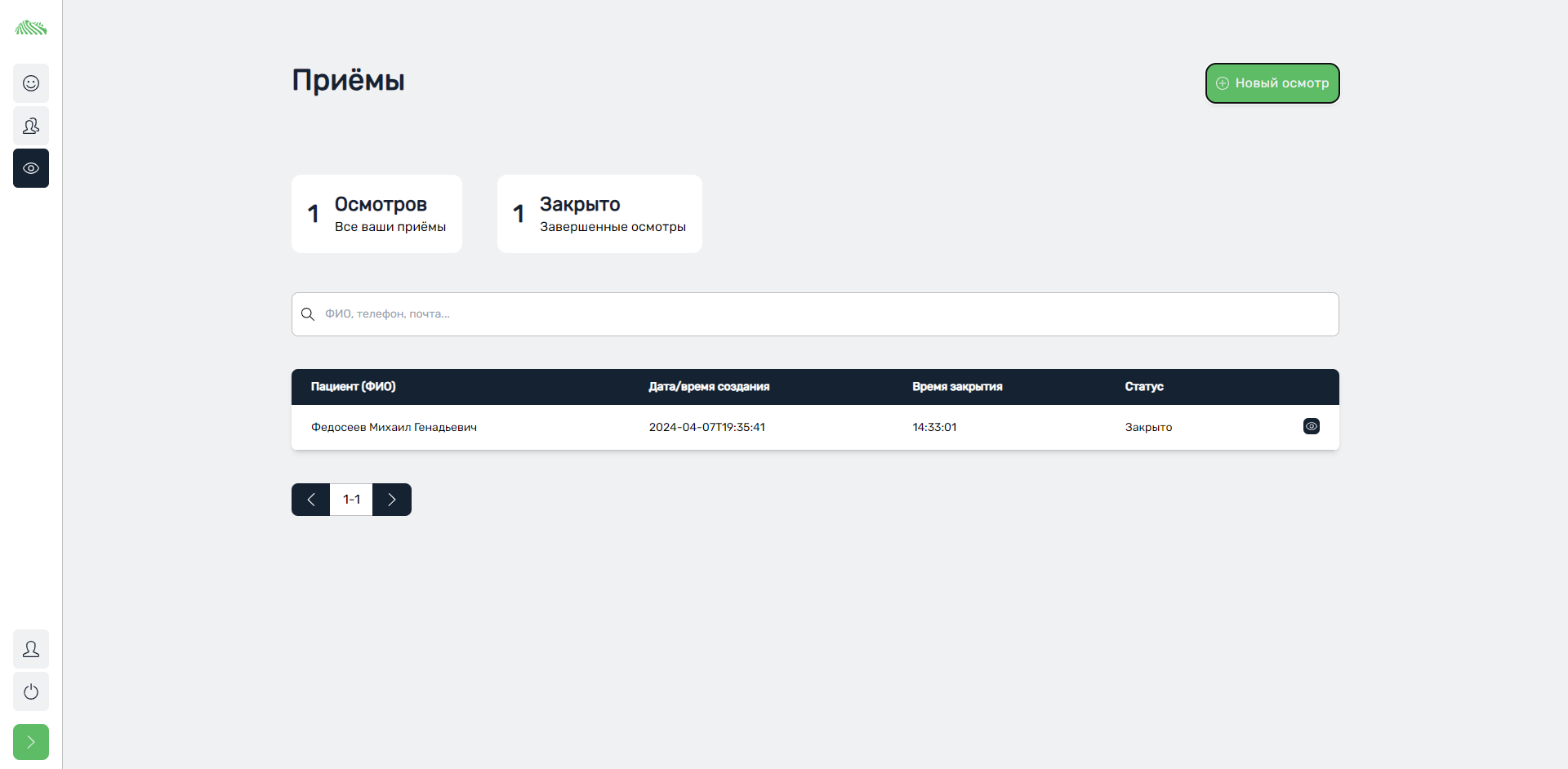
***Рисунок 30 – верстка формы создания нового пользователя***

На Рисунке 31 изображен элемент карточки конкретного пользователя.



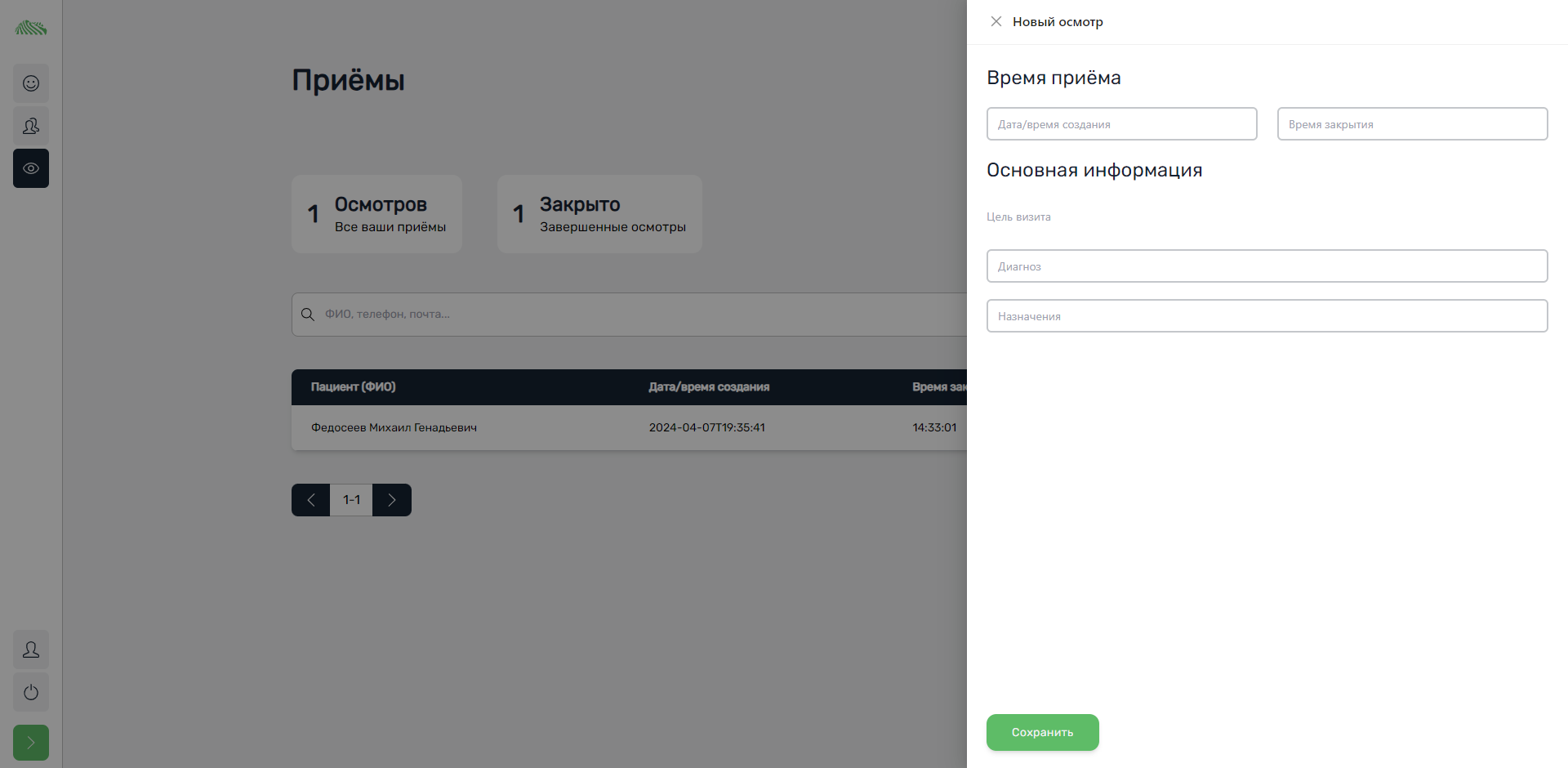
***Рисунок 31 – верстка элемента карточки пользователя***

На Рисунке 32 изображена страница приёмов (осмотров).



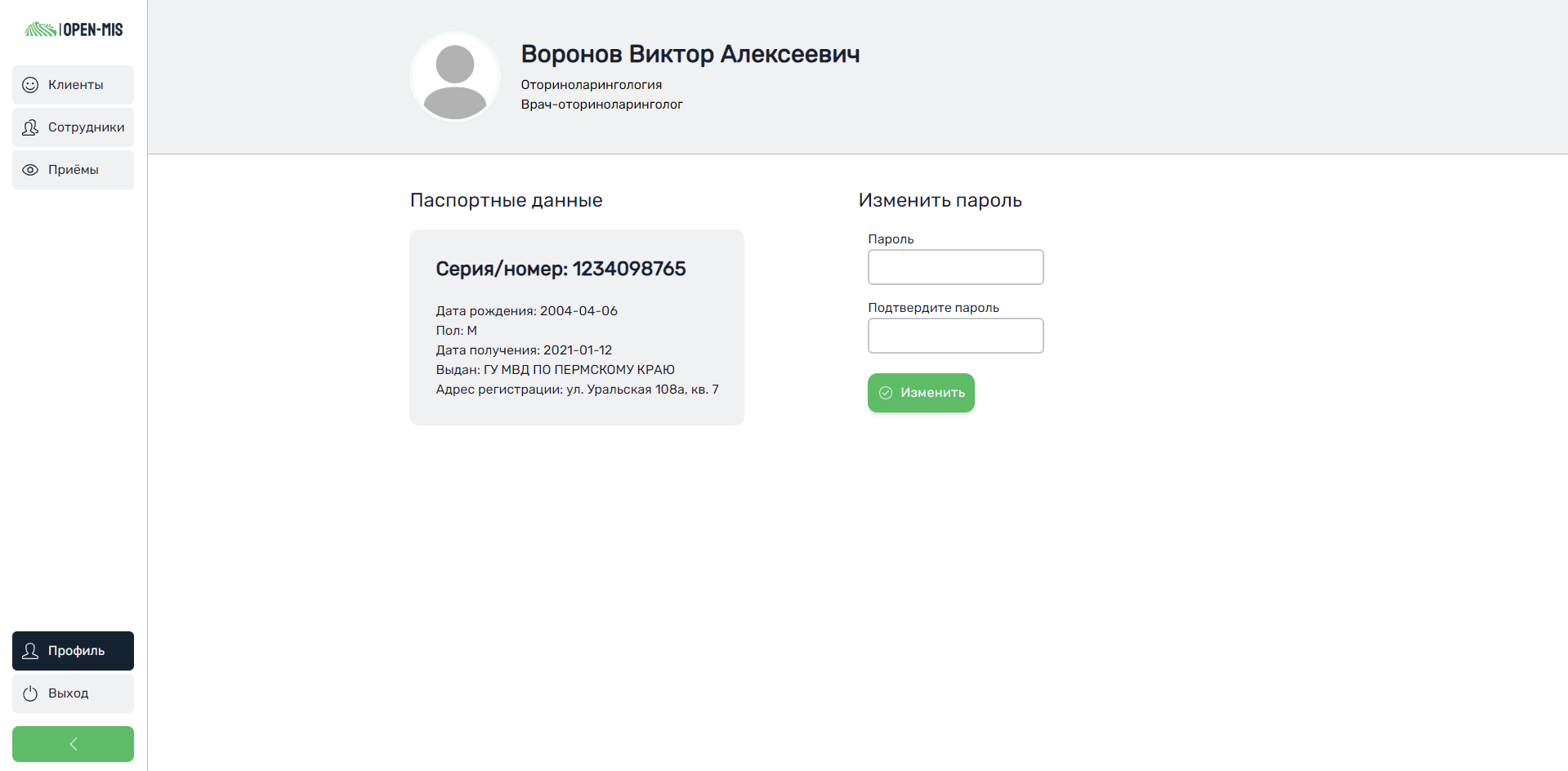
***Рисунок 32 – верстка страницы осмотров***

На Рисунке 33 изображена форма создания нового осмотра.



***Рисунок 33 – верстка формы создания нового осмотра***

Также необходимо уточнить, что реализовано выдвижное левое меню, которое сжимает элементы по правую сторону от себя. Пример представлен на Рисунке 34.



***Рисунок 34 – верстка бокового меню***

# **Глава 4. Тестирование**

Тестирование играет решающую роль в процессе создания программной системы, позволяя выявлять возможные ошибки и недочеты. В данном проекте за тестирование серверной части отвечает библиотека «pytest».

Тестирование клиентской части проводилось в формате функционального тестирования. Разработчик после написания какого-либо компонента проверял его работоспособность сначала на крайних, потом на случайных значениях.

# **Глава 5. Развертывание**

Для небольших проектов, которым требуется ограниченное количество оперативной памяти и дискового пространства, часто используется аренда части реального физического сервера, известного как виртуальный выделенный сервер (VPS). При этом сервер не принадлежит только данному проекту — его ресурсы могут быть разделены между несколькими компаниями, но каждый VPS изолирован друг от друга и не имеет доступа к данным других VPS на том же сервере. Вместо этого компании часто предпочитают выбирать отдельные выделенные серверы, хотя это и значительно дороже, чтобы полностью настраивать их под свои нужды и обеспечивать высокий уровень безопасности.

Для данного проекта было принято решение использовать упрощенный вариант развертывания, обходя стороной ручную конфигурацию веб-сервера. Был выбран бесплатный сервис Render, на котором можно выделить минимальное количество ресурсов и разместить свой виртуальный сервер [9].

Для развертывания проекта была использована технология виртуальной контейнеризации – Docker и технология группировки контейнеров Docker-compose [10].

**Заключение**

В результате разработки медицинской информационной системы был создан полноценный программный продукт, который соответствует всем описанным во введении нормам МИС и предлагает оптимальные решения для закрытия проблемных вопросов данного проекта.

**Список литературы**

1. Wikipedia. Медицинская информационная система. [Электронный ресурс]. – URL: <https://ru.wikipedia.org/wiki/Медицинская_информационная_система> – (Дата обращения 01.10.2023).
2. Pmt. Медиалог. [Электронный ресурс]. – URL: <https://medialog.ru> – (Дата обращения 01.10.2023).
3. 1С Консалтинг. МИС на базе 1С. [Электронный ресурс]. – URL: <https://consulting.1c.ru/med> – (Дата обращения 01.10.2023).
4. Tiangolo. FastAPI framework. [Электронный ресурс]. – URL: <https://fastapi.tiangolo.com/> – (Дата обращения 03.12.2023).
5. JetBrains. Python Developers Survey 2020 Results. [Электронный ресурс]. – URL: <https://www.jetbrains.com/lp/python-developers-survey-2020/> – (Дата обращения 09.01.2024).
6. JetBrains. Python Developers Survey 2022 Results. [Электронный ресурс]. – URL: <https://lp.jetbrains.com/python-developers-survey-2022/#FrameworksLibraries> – (Дата обращения 09.01.2024).
7. LOGO. Free logo. [Электронный ресурс]. – URL: <https://logo.com/> – (Дата обращения 12.02.2024).
8. Google Fonts. Rubik. [Электронный ресурс]. – URL: <https://fonts.google.com/specimen/Rubik?query=rubik> – (Дата обращения 12.02.2024).
9. Render. Render. [Электронный ресурс]. – URL: <https://render.com/> – (Дата обращения 05.04.2024).
10. Docker. Docker. [Электронный ресурс]. – URL: <https://www.docker.com/> – (Дата обращения 10.04.2024).