Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и Информационных технологий
институт
Информационные системы
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой ИС
_____ Л.С. Троценко
подпись инициалы, фамилия
« 13 » июня 2018 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 Информационные системы и технологии

Информационная система ведения электронных медицинских карт пациентов

Руководитель _____ И.А. Легалов
подпись, дата инициалы, фамилия

Выпускник _____ В.А. Малышева
подпись, дата инициалы, фамилия

Нормоконтролер _____ Ю.В. Шмагрис
подпись, дата инициалы, фамилия

Красноярск 2018

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и Информационных технологий
институт
Информационные системы
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой ИС
_____ С.А. Виденин
подпись инициалы, фамилия
« 2 » марта 2018 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту: Малышевой Варваре Андреевне

Группа: КИ14-13Б Направление: 09.03.02 «Информационные системы и технологии»

Тема выпускной квалификационной работы: Информационная система ведения электронных медицинских карт пациентов.

Утверждена приказом по университету ВКР: 4896/с от 05.04.2018 г.

Руководитель ВКР: И.А. Легалов, к.т.н., доцент кафедры «Информационные системы» ИКИТ СФУ.

Исходные данные для ВКР: системы управления медицинскими картами пациентов, список требований к разрабатываемой системе, документация к инструментам разработки, справочные данные сети Internet сайтов, материалы с преддипломной практики.

Перечень разделов ВКР: Описание предметной области, обзор существующих аналогов ИС, анализ задачи и требований, инструментальные средства разработки, проектирование архитектуры приложения, разработка приложения.

Перечень графического материала: Презентация, выполненная в Microsoft Office Point 2010.

Руководитель ВКР _____ И.А. Легалов
(подпись)

Задание принял к исполнению _____ В.А. Малышева
(подпись)

«2» марта 2018 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Информационная система ведения электронных медицинских карт пациентов» содержит 55 страниц текстового документа, 31 иллюстрацию, 18 использованных источников.

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, ИНФОРМАЦИОННЫЕ СИСТЕМЫ, DJANGO, ПРОГРАММНЫЙ ПРОДУКТ, БАЗА ДАННЫХ, СУБД, POSTGRES.

Объектом исследования является поликлиника.

Предметом исследования является введение и учет медицинских карт пациентов в поликлинике.

Целью данной работы является повышение эффективности работы поликлиники, упрощение работы медперсонала по обслуживанию пациентов.

Основные задачи:

1. Провести анализ предметной области и программных средств.
2. Разработать бизнес-логику и интерфейс.
3. Реализовать проект с использованием программных средств.
4. Протестировать программный продукт.

Основные задачи:

1. Обосновано решение о создании веб приложения для поликлиник по управлению данными о пациентах на основе анализа существующих информационных систем.
2. Создано веб-приложения для управления медицинскими картами пациентов.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность выбранной темы обусловлена тем, что на сегодняшний день электронный документооборот все больше интегрируются в различные системы. В том числе и в медицинские учреждения.

Медицинские карты пациентов, внутренние документы реально и доступно преобразовывать в электронный вид. Таким образом, регистратура любой поликлиники в считанные минуты может найти сведения о принятых пациентах, вызовах, кабинетах.

Существующие системы на сегодняшнее время не являются универсальными. Большая часть из них является слишком дорогостоящими, либо удовлетворяют не все требованиям.

Цель бакалаврской работы исследование предметной области и разработка программного приложения.

Требуется создание полнофункциональной информационной системы, использование которой будет способствовать повышению эффективности работы поликлиники, переходу на качественно новый уровень обслуживания и лечения пациентов. Также система не должна быть слишком дорогой в разработке и в обслуживании.

Для достижения цели ставятся следующие задачи:

- исследовать аналоги информационных систем данной предметной области;

- провести анализ задачи;

- определить требования к разрабатываемой системе;

- определить инструментальные средства разработки;

- разработать базовый интерфейс веб приложения;

Система логически делится на две части – для пациентов и для врачей. Врачи могут управлять данными, пациенты – получать доступ к электронной версии медицинских карт.

Прохождение лечения пациента в поликлинике подразумевает:

- оформление личной карточки пациента;

- хранение личной карточки пациента, историй всех болезней, поставленных диагнозов, результатов проведения исследований / анализов;

- направление пациента к врачу, на проведение исследования, сдачу анализов;

- оформление справок / больничных листов.

Разрабатываемая система должна отвечать задачам:

- упрощение доступа к персональным данным пациента;

- предоставление пациента доступа со своим личным данным;

- быстрое доведение результатов проведения анализов пациента до лечащего врача;

- сокращение штатной численности отдела регистратуры, расходов на зарплату и сокращение людских и временных затрат на обработку информации;

- централизованное хранение всех данных о пациенте;

- уменьшение количества противоречивых данных.

1 Общие сведения

1.1 Описание предметной области

Поликлиника - лечебно-профилактическое учреждение, оказывающее специализированную медицинскую помощь больным и больным дома.

Клиника отличается от диспансеров большим объемом деятельности и более специализированной медицинской помощью (количество специализированных и вспомогательных помещений для оказания квалифицированной медицинской помощи в клинике и многое другое).

Учреждения состоят из многих отделений - от терапевтического до функциональной диагностики, в зависимости от вида поликлиники.

Поликлиника обеспечивает:

- медицинское и неотложное лечение острой и внезапной болезни, травм и отравлений;

- медицинское лечение амбулаторных пациентов и пациентов дома с сохранением в необходимых случаях госпитализации;

- проведение анти эпидемических мероприятий (вакцинация, идентификация инфекционных заболеваний, динамический мониторинг лиц, которые были в контакте с инфекционными пациентами и выздоравливающие, тревожные, при необходимости, органы власти и др.).

Работа поликлиники основана на диспансерном методе (подбор лиц, подлежащих динамическому мониторингу, специализированному лечению, систематическому изучению условий труда и жизни населения). В клинике проводятся профилактические мероприятия, которые в сочетании с медицинской работой являются основными целями поликлиник и играют решающую роль в сокращении заболеваемости среди населения.

Поликлиника - это динамический контроль состояния здоровья населения.

В условиях работы поликлиники приходится иметь дело с большими потоками информациями, управлять и изменять данные о пациентах, врачах, страховых компаниях и многое другое.

1.2 Обзор аналогов ИС

Рассмотрим самые популярные информационные системы для поликлиник в Красноярском крае.

1.2.1 Информационная система 1С: Медицина

1С: Медицина. Поликлиника – система для учета сложных и комплексных услуг ведение программ медицинского обслуживания в решениях линейки 1С.

1С это один из крупнейших отечественных разработчиков ПО для именно коммерческих и государственных предприятий, структур и организаций

Это подразумевает то, что платформа работает достаточно стабильно. У платформы есть огромная поддержка большая аудитория пользователей так и в программистов, руководителей, ее могут доработать или написать для вас практически с нуля.

Программа может контролировать доступ отдельных пользователей к различным частям информационной базы, чтобы не предоставить им больше того.

Есть различные механизмы обмена данными между конфигурациями 1С, обмена данными с оборудованием, с анализаторами. При задействовании программистов можно обменяться данными с чем угодно и это задействовать эти данные в своей учетной системе.

Конфигурация:

- регистрацию и учет пациентов;
- видений сведения пациентах, включая персональные данные, контактную информацию;
- управлением данными о медицинском страховании;
- управление взаимоотношения с контрагентами;

- введение сведений о контрагентах, страховых медицинских организациях и других юридических и физических лицах оплачивающих медицинские услуги;

- регистрация и ведение договоров на оказание медицинских услуг;

- управление про правилами продаж медицинских услуг;

- назначение цен и скидок проведения маркетинговых акций и регистрации;

- контроль ценовых, финансовых, объемно-календарных медицинских услуг;

- планирование работы медицинской организации;

- ведение графиков работы медицинского персонала;

- назначение места, времени и продолжительности оказания медицинских услуг пациентам;

- прием оплаты за медицинские услуги ¬ учет наличных и безналичных денежных средств, принимаемых от пациентов счет оплаты оказываемых услуг печать кассовых чеков;

- учет оказанных медицинских услуг и ведения электронной медицинской карты (учет диагнозов, назначенных медицинских услуг и лекарственных средств);

- учет взаимодействии с клиентами;

- учета взаимодействия с физическими и юридическими лицами по телефону и электронной почте.

Также в системе есть многие полезные функции.

- возможность создания колл-центра с возможностью интеграции с различными телефонными системами;

- запись, перезапись, отмена предварительной записи на прием по телефону;

- автоматическое определение пациента по номеру телефону или электронной почте;

- быстрый и удобный набор исходящих телефонных звонков гражданам для уведомлений пациентов о предстоящем визите;

- быстрый и удобный набор исходящих телефонных звонков гражданам для уведомлений пациентов о предстоящем визите;

- уведомление об окончании срока временно нетрудоспособности;

- информирование пациентов о регистрации запись к врачу перенос по каким-либо причинам времени посещения врача;

Возможности информационно-аналитической службы:

- закрытие графиков работы врачей на рабочие места при наличии записанных пациентов;

- в справочнике рабочие места добавлен реквизит время выполнения, если у элемента справочника этот реквизит будет заполнен, то все услуги на данном рабочем месте выполняются с указанным временем;

- для подразделения организации добавлена возможность поместить в архив;

Пример интерфейса изображен на рисунке 1.

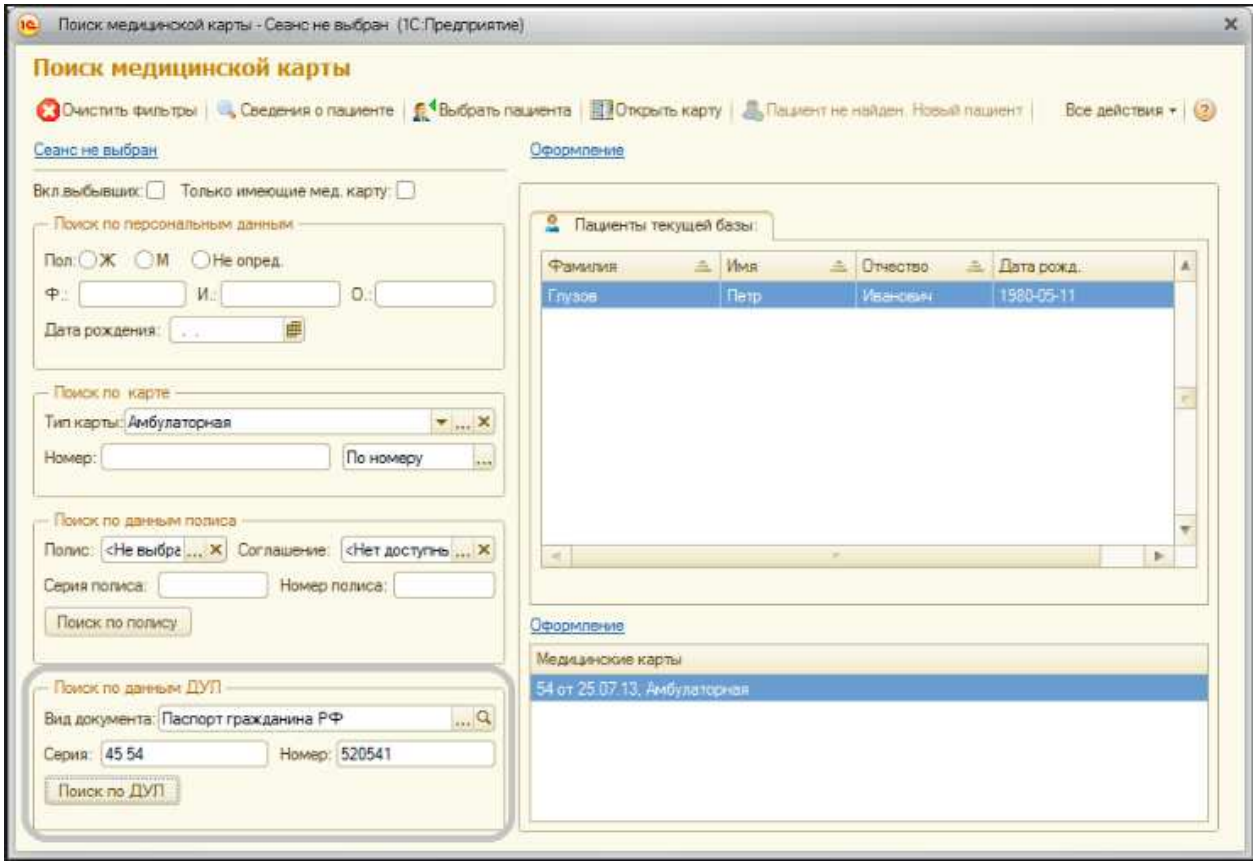


Рисунок 1 – Интерфейс ИС 1С: Медицина: Поликлиника

Плюсы рассматриваемой информационной системы:

- есть возможность разделять данные;
- широкий выбор фильтров;
- настройка оповещения пациентов по электронной почте и телефону;
- можно ввести несколько медицинских карт для одного пациента;
- в одной базе можно вести несколько организаций;
- поддержка тонкого клиента для операционной системы linux;
- техническая поддержка продукта.

Минусы:

- Не интуитивный интерфейс;
- для внедрения в небольшую поликлинику система слишком тяжеловесна и трудоемка в использовании;
- стоимость (28 тыс. рублей будет стоить только покупка лицензии, в дальнейшем необходимо продлевать лицензию и оплачивать поддержку).

- нет интерфейса для пациентов.

1.2.2 Electronic Medical Records Software от компании MicroMD

MicroMD - это облачная система управления практикой и электронная медицинская документация (EMR), предназначенная для медицинских работников. Он содержит инструменты, помогающие пользователям выполнять повседневные операции более эффективно, чтобы они могли преуспеть в своей специализации.

Продукт глобального поставщика медицинских услуг Генри Шейн, MicroMD имеет комплексные решения для управления практикой для специалистов всех специальностей и размеров клиник.

Расширенные функции для планирования, регистрации пациентов, управления персоналом, интегрированных решений EDI, детальной отчетности, форматирования электронного файла заявок и т.д. Как заявляет производитель, все сделано так, чтобы позволить пользователям максимально оптимизировать рабочие процессы. Система проста в освоении и может быть масштабирована для поддержки растущих потребностей в клинике.

ПО подходит для широкого круга медицинских направлений, но может не подходит для некоторых специальностей и типов организации. На данный момент разработчики продолжают расширять и создавать контент и функциональные возможности EMR для удовлетворения различных потребностей в специальностях каждый год.

Как заявляют разработчики, программа помогает уменьшить ненужную бумажную волокиту, повышает производительность клиники и уровень обслуживания клиентов.

Основные характеристики MicroMD EMR:

- Электронные медицинские карты.

- Поддержка E/M кодинга (актуально для США, стандарт ведения медицинских карт и обслуживания страховых компаний).

- Совместимость с HIPAA (Закон о переносимости и подотчетности медицинского страхования).

- Систему EMR можно настроить под себя.

- Настраиваемая клиническая документация.

- Поддержка многих ОС.

- Электронная подпись.

- Просмотр и сохранение диагностических и анатомических изображений.

- Поддержка пациентов, оповещения и напоминания о записях к врачам.

- Управление изображениями и аннотациями.

- Управление рабочей нагрузкой.

- Встроенный административная панель.

- Есть настройка взаимодействия с аптеками (доставка лекарств, выписка рецептов)

- Открытая архитектура и взаимодействия с системой.

- Подключение устройств.

- Интерфейс с цифровыми мониторами сердечно-легочной и жизненно важных функций.

- Использовать стандартную и настраиваемую отчетность для анализа тенденций.

- Позволяет настроить вашу электронную медицинскую систему записи.

- Поддержка отчетности.

- Специальные системные продуманные шаблоны.

- Управление записью к врачам.

- Удобный просмотр клинических данных пациента.

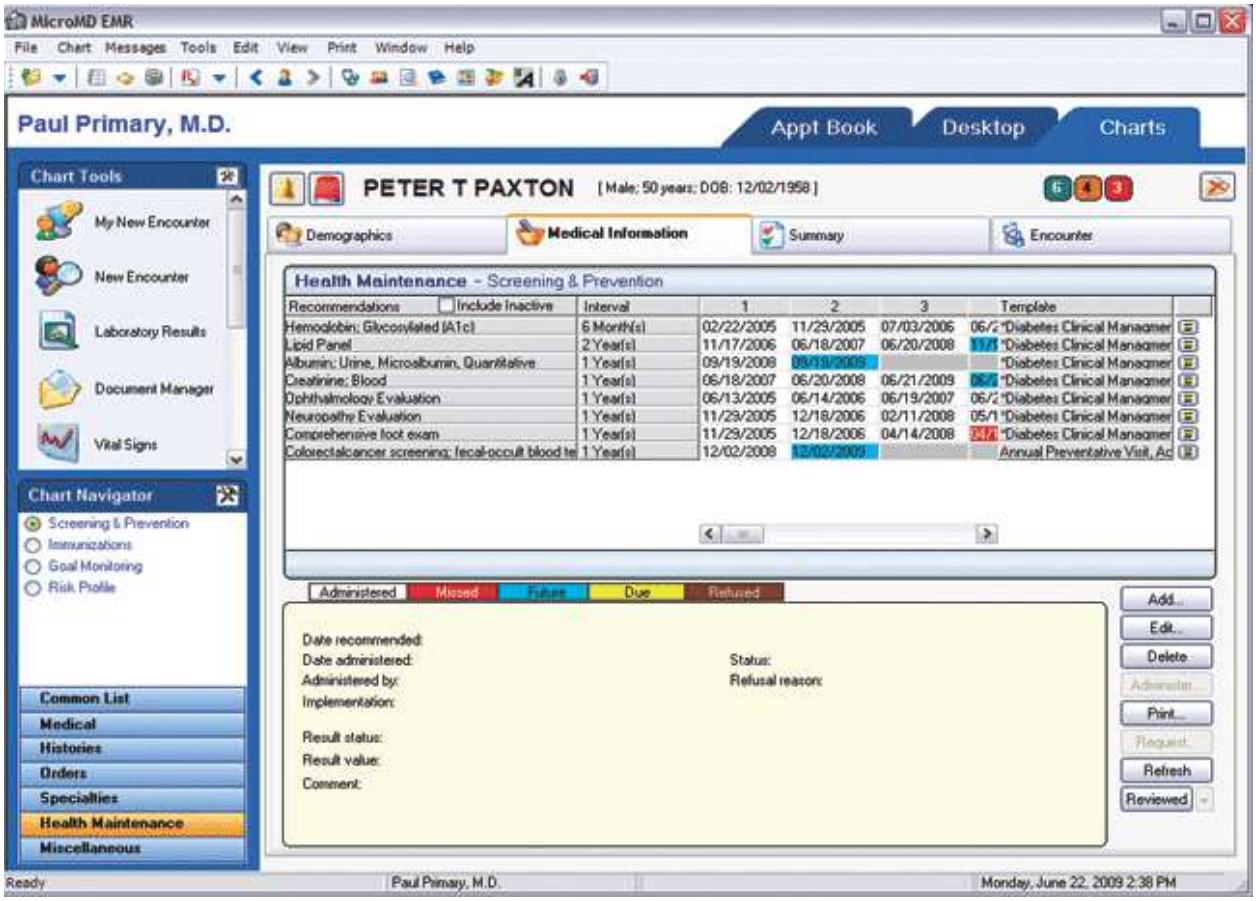


Рисунок 2- Интерфейс EMR

Плюсы подсистемы:

- огромные функциональные возможности;
- хороший интерфейс;
- поддерживается законодательные стандарты (США);
- просмотр расписания на сайт.

Минусы:

- стоимость интеграции и поддержки - $545 в месяц;
- ориентирована на стандарты американского здравоохранения;
- поддерживается только английский язык;
- подходит не всем, непонятно как выяснить кому именно;
- только декстопная версия.

1.2.3 Сравнение аналогов

ИС«1С: Медицина. Поликлиника» достаточно мощная и функциональна, хорошо ориентирована на российскую медицинскую инфраструктуру. Многие клиники пользуются этой системой. Но в ней есть и свои недостатки, нет никакой поддержки UI для пациента. Интерфейс устаревший. Не подходит для других стран.

«Electronic Medical Records» еще более масштабная и функциональная чем российский аналог. Но эта ИС ориентирована на западный рынок медицинских услуг, поддерживает только один язык и очень дорогая.

1.2.4 Выводы

Вышеперечисленные системы имеют свои преимущества и минусы. Но ни одна их них не имеет достаточно гибкости к настройке, нет интерфейса для пациентов и нет поддержки веб-версии. Было принято решение реализовать свой аналог системы управления медицинскими картами пациентов, используя современные веб-технологии.

1.3 Анализ задачи

1.3.1 Требования к функционалу приложения

По результату сделаны следующие выводы о требованиях веб-приложения:

- использует современные языки и фреймворки для продолжительной поддержки;
- быть легко страиваемым и модульным;
- иметь бэкенд часть (для медицинских работников) и фронтенд часть (для пациентов);

- поддерживать необходимый уровень защиты от хакерских атак и кэширование;

- малотребовательным к программным и аппаратным средствам;

Основной особенностью данной системы будет доступность и малобюджетная разработка. Также систему смогут легко поддерживать сторонние разработчики. ИС пригодна для внедрения множеств функциональных настроек, может легко масштабироваться.

Данная система – это прототип для демонстрации того, каким образом можно будет использовать систему на реальных задачах.

1.3.2 Инструментальные средства для разработки

Python

Python — объектно-ориентированный язык. Программные модули и структуры данных могут использоваться как объекты, т.е. имеют свойства и методы.

Подпрограммы в Python оформляются только как функции, но эти функции могут не возвращать значений, а могут возвращать несколько значений в виде структур данных.

Функции, оформленные по определённым правилам, объединяются в модули (библиотеки функций). Модули (или некоторые функции из модулей) по мере необходимости подключаются к пользовательским программам. Такой подход позволяет экономить память вычислительной системы и не занимать её ненужным кодом.

Философия Python включает в себя такие правила:

- Красивое лучше, чем уродливое.

- Явное лучше, чем неявное.

- Простое лучше, чем сложное.

- Читаемость имеет значение.

- Должен существовать один - и, желательно, только один - очевидный способ сделать это.

И язык Python вполне соответствует этим постулатам. Простой, ясный с первого взгляда, непротиворечивый синтаксис, использование традиционной алгебраической записи и привычных по другим языкам конструкций, простые средства для работы со сложными типами данных, строгая модульность и компактность кода, богатые возможности объектно-ориентированного программирования - все это присутствует. А если добавить сюда еще и высокое (разумеется, для интерпретируемого языка) быстродействие, богатейшую стандартную библиотеку и наличие огромного числа сторонних библиотек, выполняющих самые разные задачи, становится совсем не удивительно, что Python в настоящее время столь популярен.

С помощью этого языка можно писать настольные приложения, в том числе имеющие графический интерфейс, системные утилиты, интернет-приложения и командные сценарии для других программных пакетов. А еще на Python можно разрабатывать Web-серверные приложения- например, Web-сайты.

Этот язык выбран в качестве основного для разработки, так как достаточно просто можно создавать сложны, легко поддерживаемые приложения. И одно из самых лучших достоинств – развитое сообщество. Это означает, что качественных фрейворков, сторонних библиотек, туториалов и решения сложных задач предостаточно.

Django web framework

Django - это высокоуровневая веб-инфраструктура Python, которая позволяет быстро создавать безопасные и поддерживаемые веб-сайты. Построенный опытными разработчиками, Django заботится о многих проблемах веб-разработки, поэтому вы можете сосредоточиться на написании своего приложения без необходимости изобретать велосипед. Это бесплатный и

открытый источник, имеет процветающее и активное сообщество, отличную документацию и множество опций для бесплатной и платной поддержки.

Использует шаблон проектирования MVC.

Django - это высокоуровневая веб-инфраструктура Python, которая позволяет быстро создавать безопасные и поддерживаемые веб-сайты. Построенный опытными разработчиками, Django заботится о многих проблемах веб-разработки, поэтому вы можете сосредоточиться на написании своего приложения без необходимости изобретать велосипед. Это бесплатный и открытый источник, имеет процветающее и активное сообщество, отличную документацию и множество опций для бесплатной и платной поддержки.

Стоит добавить, что фреймворк имеет хорошую поддержку со стороны сообщества разработчиков и наличие в Интернете большого количества сайтов, посвященных Django, на которых можно найти статьи по программированию, ответы на часто возникающие вопросы и готовые примеры кода.

Некоторые возможности и особенности Django:

- реализация архитектуры «модель-контроллер-шаблон»;

- включает в себя легкий и автономный веб-сервер для разработки и тестирования;

- реализация принципа DRY (Don't Repeat Yourself, не повторяйся), в результате;

- присутствует система сериализации и проверки формы, которая может переводить между HTML-формами и значениями, подходящими для хранения в базе данных;

- система интернационализации, включая перевод собственных компонентов Django на различные языки;

- поддержка классов промежуточного программного обеспечения (“middleware”), которые могут вмешиваться на разных этапах обработки запросов и выполнять пользовательские функции;

- структура кэширования, которая может использовать любой из нескольких методов кеширования;

- интерфейс к встроенной платформе тестирования Python;

- мощный шаблонизатор, основанный на специальных тегах, с возможностью наследования шаблонов;

- унифицированные средства для работы с базами данных любых поддерживаемых форматов: SQLite, MySQL, PostgreSQL, Oracle, Microsoft SQL Server, Firebird и др;

- авторизация и аутентификация, подключение внешних модулей аутентификации;

- динамический административный интерфейс;

- богатые средства для работы с формами;

- инструменты для реализации разграничения доступа;

- встроенный административный сайт с возможностями настройки, который можно использовать для работы с данными;

- простая и ясная структура создаваемых сайтов: каждый раздел сайта представляет собой отдельное приложение, которое может быть отчуждено и использовано в другом сайте; все приложения объединяются в проект, собственно, и представляющий собой сайт.

Стоит отметить, что такие известные сайты, как YouTube, DropBox, Reddit написаны на основе данного фреймворка.

Наряду с преимуществами, приходят недостатки. Существует много недостатков Django, упомянутых ниже:

- Использует шаблон маршрутизации, указывая его URL-адрес.

- Django слишком монолитен.

- Все основано на Django ORM.

- Компоненты развертываются вместе.

- Развивается не так быстро, как другие фреймворки.

Несмотря на его недостатки, именно этот фреймворк выбран для разработки приложения. На его основе можно преобразовать простой web сайт в мощное, и при этом легко поддерживаемое приложение.

PostgreSQL

PostgreSQL – реляционная система управления базами данных, то есть основана на теории множеств, реализована в виде двумерных таблиц, где данные хранятся по строкам, и строго контролирует типы столбцов. Несмотря на растущий интерес к новым тенденциям в области баз данных, реляционный стиль является самым популярным и, вероятно, останется таким еще довольно долго.

В PostgreSQL встроена поддержка Unicode, последовательностей, наследования таблиц, подзапросов, а реализация SQL точнее следует стандарту ANSI, чем любая другая из представленных на рынке. СУБД является быстрой и надежной, способна хранить терабайты данных и доказала работоспособность в таких высоконагруженных проектах, как Skype, французская Национальная касса по выплате пособий многодетным семьям (Caisse Nationale d'Allocations Familiales – CNAF) и Федеральное управление гражданской авиации США.

СУБД поддерживает большую часть стандарта SQL и предлагает множество современных функций:

- сложные запросы;
- внешние ключи;
- триггеры;
- изменяемые представления;
- транзакционная целостность;
- многоверсионность.

Кроме того, пользователи могут всячески расширять возможности PostgreSQL, например, создавая свои:

- типы данных;
- функции;
- операторы;
- агрегатные функции;
- методы индексирования;

- процедурные языки.

А благодаря свободной лицензии, PostgreSQL разрешается бесплатно использовать, изменять и распространять всем и для любых целей — личных, коммерческих или учебных.

Redis

Redis (REmote DIctionary Service – удаленная служба словарей) – простое в работе хранилище ключей и значений с развитым набором команд. Первая версия системы вышла в 2009 году. И в быстродействии у него практически нет соперников. Чтение производится быстро, а запись еще быстрее – на некоторых эталонных тестах продемонстрировано до 100 000 операций SET в секунду. Создатель Redis Сальваторе Санфилиппо (Salvatore Sanfilippo) называет свой проект «сервером структур данных», чтобы подчеркнуть заложенные в него возможности работы со сложными типами данных и другие особенности.

Используется для лучшего функционирования базы данных Postgres.

Помимо сервера структур данных, Redis является еще и блокирующей очередью (или стеком), а также системой публикации-подписки. В ней можно настраивать политики срока хранения, уровнидолговечности и параметры репликации. Все это делает Redis скорее комплектом инструментальных средств, в который входят полезныеалгоритмы работы со структурами данных и процессы, нежели базой данных какого-то определенного жанра.

Docker

Docker - это приложение, позволяющее легко и быстро запускать другие приложения и процессы в контейнерах (контейнеризация), которые подобны виртуальным машинам, но являются при этом более портируемыми, потребляют меньше ресурсов, а также меньше зависят от операционной системы машины-хоста.

Docker в первую очередь разработан для Linux, где он использует функции выделения ресурсов ядра Linux, такие как группы и пространства имен ядер, и файловую систему с поддержкой объединения. Поддержка ядра Linux для пространств имен в основном изолирует представление приложения для операционной среды, включая деревья процессов, сети, идентификаторы пользователей и смонтированные файловые системы. Группы ядра обеспечивают ограничение ресурсов для памяти и процессора. Начиная с версии 0.9, Docker включает библиотеку libcontainer в качестве своего собственного способа напрямую использовать средства виртуализации, предоставляемые ядром Linux.

С помощью данного инструмента работает база данных и есть возможность легко развернуть проект на сервере.

Git – система контроля версии.

Git - это свободная и открытая распределенная система контроля версий для отслеживания изменений в компьютерных файлах и координации работы над этими файлами среди нескольких людей.

Git был создан Линусом Торвальдсом в 2005 году для разработки ядра Linux. Другие разработчики ядра вносят свой вклад в его первоначальную разработку.

Инструмент в основном используется для управления исходным кодом в разработке программного обеспечения, но его можно использовать для отслеживания изменений в любом наборе файлов. Как распределенная система контроля версий, она нацелена на скорость, целостность данных, и поддержку распределенных нелинейных рабочих процессов.

Основное различие между Git и любым другим VCS (включая Subversion и друзей) - это то, как Гит думает о своих данных. Концептуально большинство других систем хранят информацию как список изменений, основанных на файлах. Эти другие системы (CVS, Subversion, Perforce, Bazaar и т. Д.) Считают

информацию, которую они хранят, как набор файлов, и изменения, внесенные в каждый файл с течением времени (это обычно описывается как управление версиями на основе изменений).

Git считает свои данные более похожими на ряд снимков миниатюрной файловой системы. С Git, каждый раз, когда вы совершаете или сохраняете состояние своего проекта, Git в основном делает снимок того, как выглядят все ваши файлы в данный момент, и сохраняет ссылку на этот снимок.

Отличительное преимущество инструментов, подобным Git является то, что каждый, кто имеет копию со всем необходимым, может восстановить проект. Если центральный сервер сломается и все данные потеряются, то любая удаленная копия может стать официальной, потому что это будет достаточно для восстановления работоспособности. Единственное расхождение только возникает, когда разработчики не имеют последние коммиты.

2 Разработка приложения

2.1 Проектирование моделей в базе данных

Веб-приложения Django получают доступ и управляют данными через объекты Python, называемые моделями. Модели определяют структуру хранимых данных, включая типы полей и, возможно, их максимальный размер, значения по умолчанию, параметры списка выбора, текст справки для документации, текст меток для форм и т. д. После того, как вы выбрана какая то база данных, нам не нужно напрямую работать с ней. Просто пишем свою структуру модели и код, а Django делает всю основную работу, связанную с базой данных за вас.

Определим перечень объектов, имеющих влияние на информационную систему, и установим между ними связи:

- Пользователи;
- Пациенты;
- Доктора;
- Администраторы;
- Медицинская карта;
- Записи в медицинской карте;
- Анализы;
- Хронические заболевания;
- Прививки.

Вспомогательные являются сущности аккаунтов (социальных сетей), email адресов, связи токенов и разрешений юзеров.

Для начала мы свяжем базу данных с самим Django. Для запуска Postgres используется и клиет, а Docker контейнер. Определяем неоюходимые настройки в doche-compose.yaml (рисунок 3).

Кэширование базы данных - широко используемый метод повышения масштабируемости. Разгружая работу базы данных в другие, более быстрые хранилища, она также может помочь улучшить доступность данных. Поэтому мы будем использовать также Redis вместе с Postgres.

Рисунок 3 – Настройки образов Postgres и Redis

Почему именно Redis? Redis - это система хранения ключевых значений, которая работает в оперативной памяти, она похожа на «легкую базу данных», и поскольку она работает на уровне памяти RAM, она на порядок быстрее, чем чтение / запись в PostgreSQL или любую другую традиционную реляционную базу данных. Redis - это так называемая база данных NoSQL, такая как Mongo и многие другие. Он не может напрямую заменить PostgreSQL, вам все равно требуется постоянное хранилище, но оно работает вместе с реляционными базами данных как альтернативная система хранения. Вы можете использовать Redis, если ваши операции ввода-вывода начнут дорожать, и это отлично подходит для быстрых вычислений и запросов на основе ключей. Главное достоинство - уменьшение количества запросов к PostgreSQL.

Запуск базы конфигурации производится через команду docker-compose up postgres redis (рисунок 4).

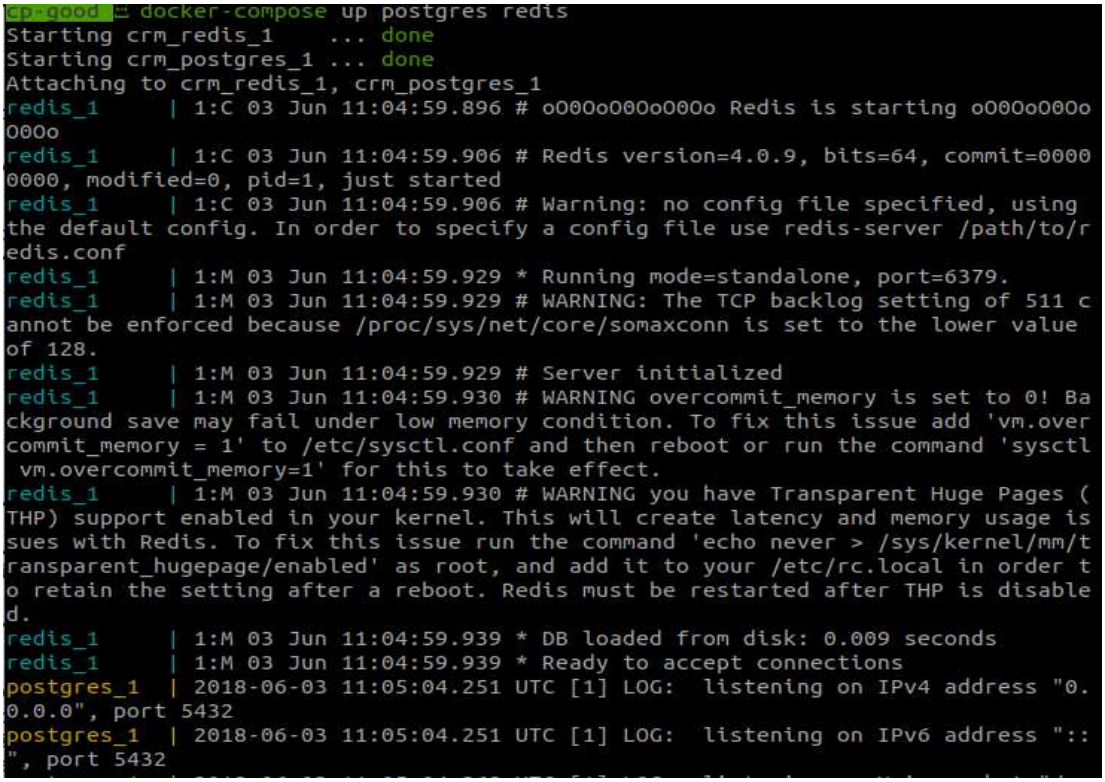


Рисунок 4- Работа образа Postgres и Redis

Теперь необходимо создать таблицы и отношения для базы данных.

Была разработана предварительная UML диаграмма классов (рисунок 2). На рисунке 5 изображены все сущности, созданные в базе данных Postgres.

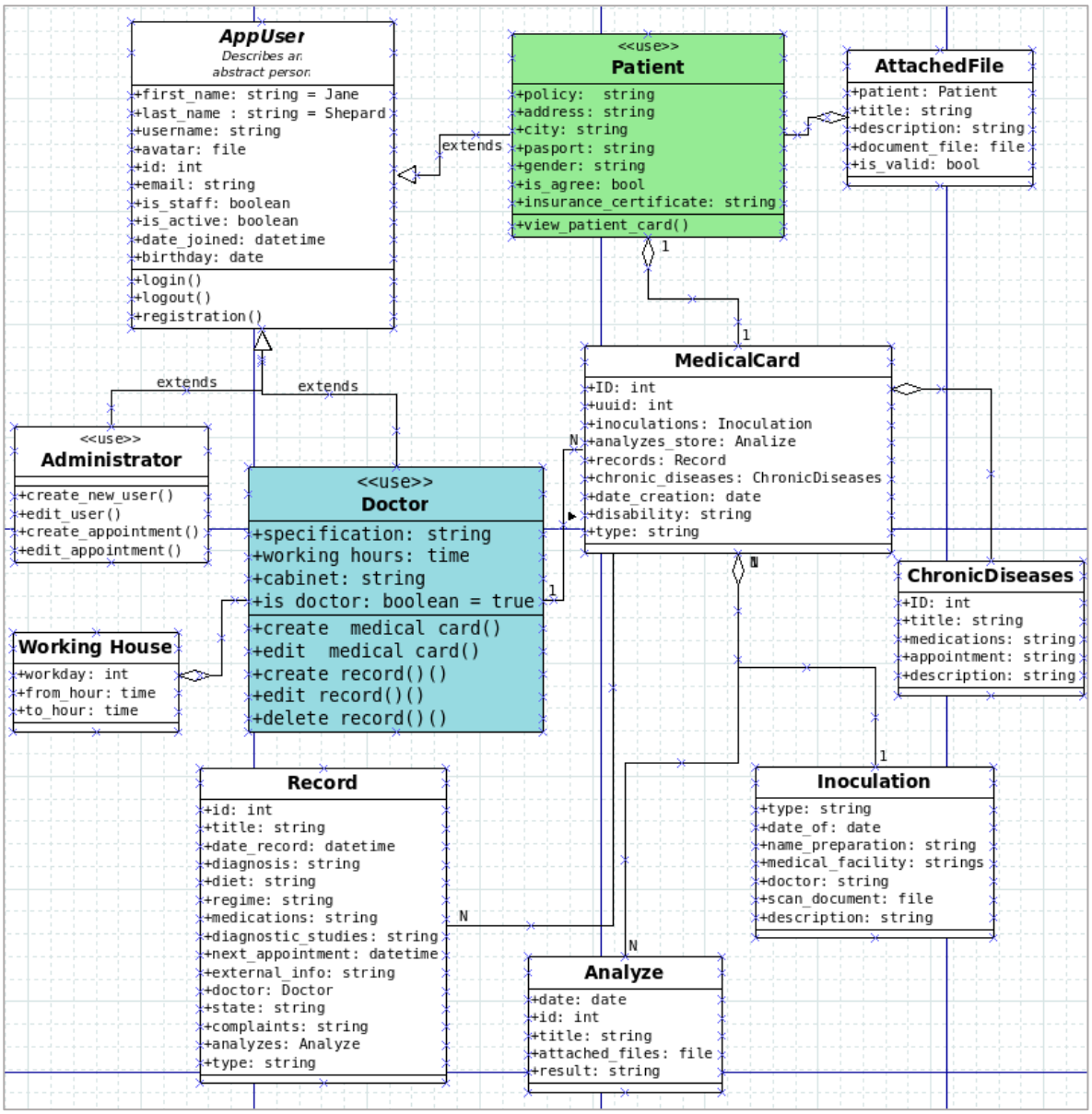


Рисунок 5- Диаграмма классов

Перечислим основные сущности системы:

- AppUser, базовая сущность пользователя;
- Patient, расширение пользователя, представляющего пациента;
- Doctor, расширение пользователя, представляет сущность доктора;
- Medical card, характеризует сущность медицинской карты пациента;
- Record, сущность записи в медицинской карте;
- Analyze, перечень анализов пациента;
- Inoculation, характеризует записи о прививках пациента;

- AttachedFile, сущность для различных файлов пациента (скан копии документов, анализов и пр.);

- Working House, сущность отображающая режим работы врачей.

Остальные сущности как EmailAddress, Account и прочее являются частями внешних библиотек, используется для расширения функций пользователя.

Сущность AppUser является базовой для всех пользователей. На рисунке 6 и 7 представлены взаимоотношения между классами.

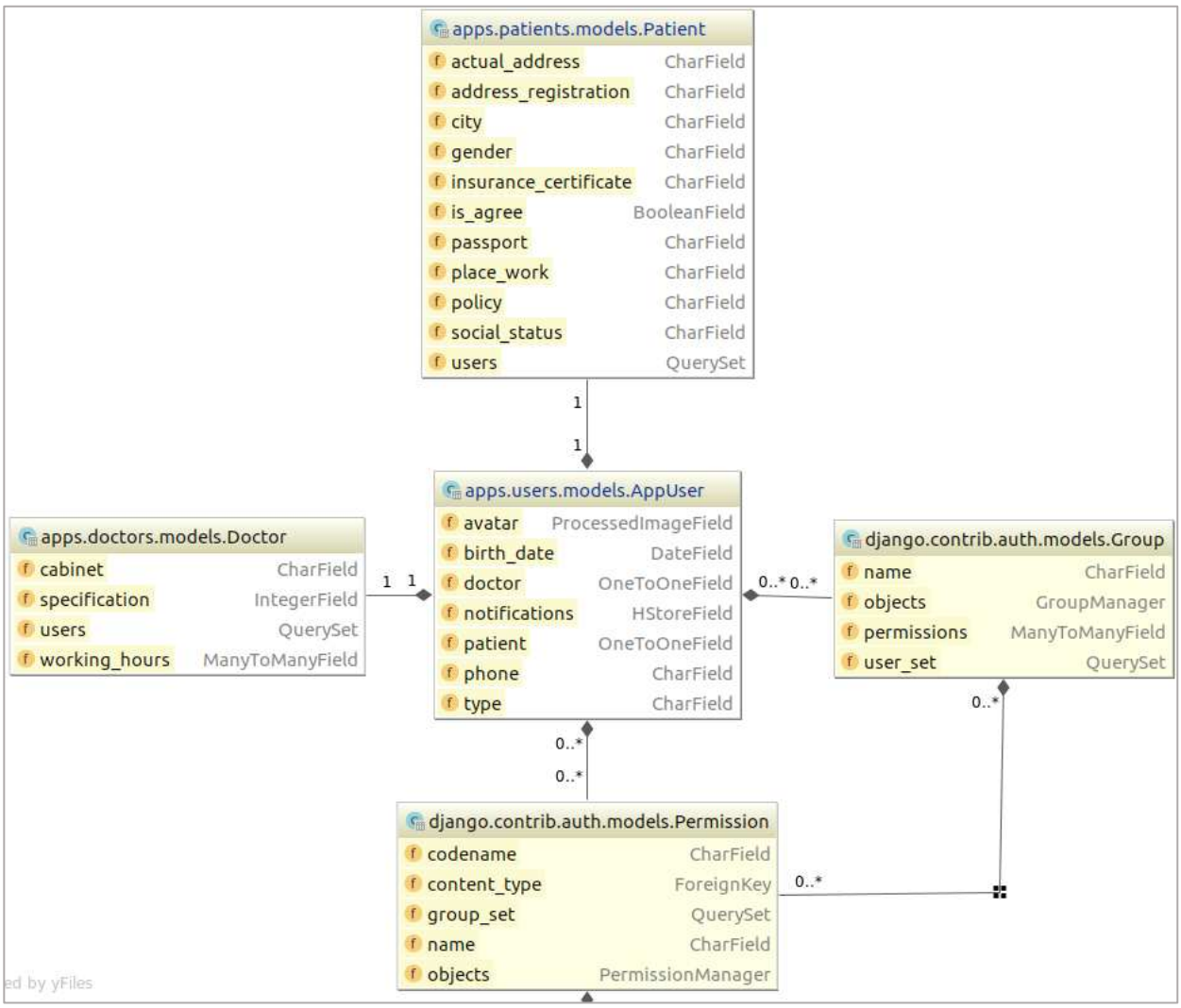


Рисунок 6 - Диаграмма связных сущностей AppUser и его расширений

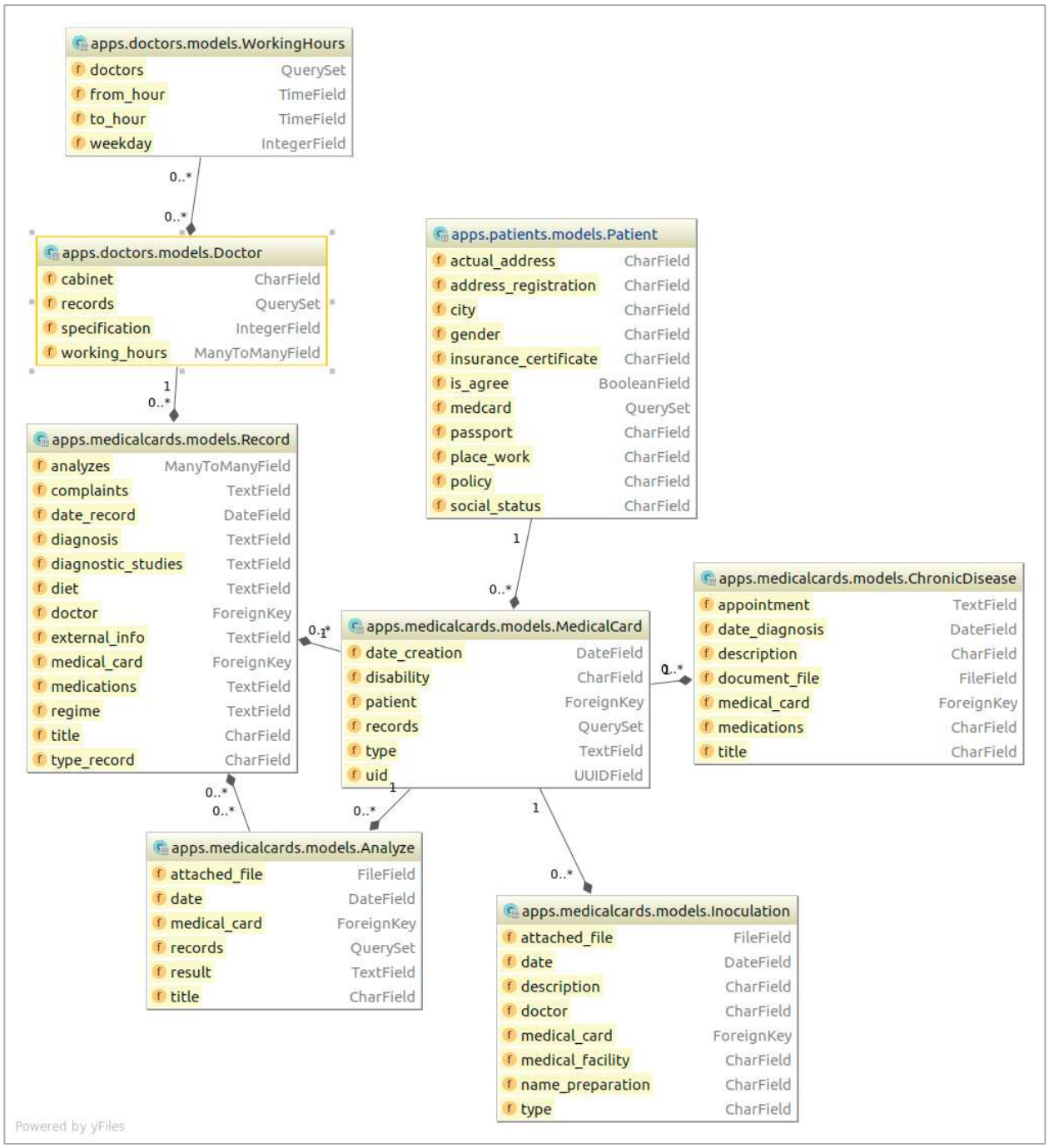


Рисунок 7 - Диаграмма связных сущностей с MedicalCards

2.2 Реализация быстрого создания тестовых данных

Для того чтобы приложение было проще тестировать, и нагляднее редактировать внешний вид, необходимы некоторые тестовые данные.

Вручную создавать пользователей, пациентов – долго и не эффективно, и особенно если между сущностями множество связей, большое кол-во полей и прочее.

Для упрщения данный задачи в проекте используются две библиотки – Factory Boy и Faker. Они помогают создать тестовые данные быстро, при этом создавая одновременно все зависимые сущности.

Foreign Key разработан, чтобы хорошо работать с различными ORM (Django, Mongo, SQLAlchemy). С Django он работает идеально – можно даже настроить единовременно создание Foreign Key сущностей. Эта библиоткека включает в себя и пакет Faker.

Faker - это пакет Python, который генерирует поддельные данные. Независимо от того, нужно ли загружать базу данных, создавать красивые XML-документы, заполнять свою настойчивость, проводить стресс-тестирование или анонимизировать данные, взятые из производственной службы.

На рисунке 8 представлен класс для создания сущности медицинской карты.

```
class MedicalCardFactory(factory.django.DjangoModelFactory):
    class Meta:
        model = MedicalCard

    uid = factory.Faker('uuid4')
    date_creation = fake_factory.date_between(start_day, today)
    disability = fake_factory.sentence(nb_words=4,
                                       variable_nb_words=True,
                                       ext_word_list=None)
    type = fake_factory.word()
    patient = factory.SubFactory(PatientFactory)

    record = factory.RelatedFactory(RecordFactory, 'medical_card')
    inoculation = factory.RelatedFactory(InoculationFactory, 'medical_card')
    analyze = factory.RelatedFactory(AnalyzeFactory, 'medical_card')
    chronic_disease = factory.RelatedFactory(ChronicDiseaseFactory,
                                             'medical_card')
```

Рисунок 8 - Создание объекта модели медицинской карты

Стоит обратить внимание, что для всех связных сущностей (Анализы, Записи, Хронических заболеваний, Прививки, Пациент) есть свои отдельные

фабрики. То есть медицинская карта создается уже с нужными объектами и связями. На рисунке 9 представлен класс создания объекта пациента.

```
import factory
import faker

from apps.patients.models import Patient

fake_factory = faker.Faker()


class PatientFactory(factory.django.DjangoModelFactory):

    class Meta:
        model = Patient

    actual_address = factory.Faker('address')
    address_registration = factory.Faker('address')
    city = factory.Faker('city')
    gender = fake_factory.random_element(['M', 'F'])
    policy = fake_factory.uuid4()
    passport = fake_factory.credit_card_number()
    insurance_certificate = fake_factory.credit_card_full()
    place_work = fake_factory.company()
    social_status = fake_factory.random_element(
        ['Student', 'worker', 'unemployed', 'pensioner'])
    user = factory.RelatedFactory('apps.users.factories.UserFactory',
                                  'patient', type='patient')
```

Рисунок 9 - Создание объекта модели Пользователь

Вместе с пациентов создается объект пользователя с нужным типом (рисунок 10). Таким же образом создается и модель Доктора.

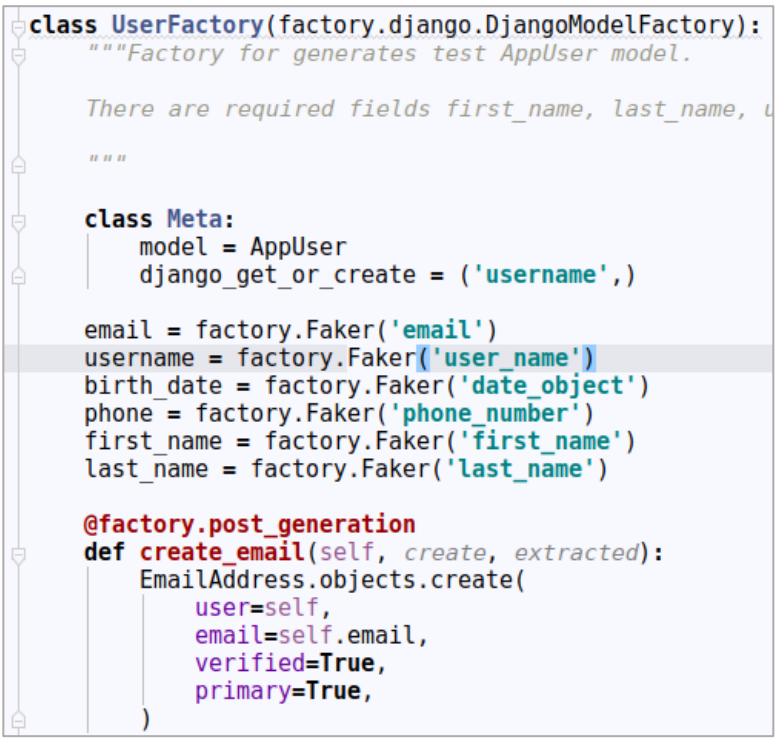


Рисунок 10 - Создание объекта модели Пользователь

Таким образом, в консоли одной строкой *"m = MedicalCardFactory()"* можно создать сколько угодно объектов пациентов, медицинских карт и прочее. Это повышает эффективность и качество тестирования приложения.

2.3 Реализация входа и выхода из системы

Система аутентификации Django обрабатывает как аутентификацию, так и авторизацию. Вкратце, аутентификация проверяет, является ли пользователь тем, кем он является, а авторизация определяет, что разрешено для аутентифицированного пользователя. Здесь термин «аутентификация» используется для обозначения обеих задач REST framework (из модуля django-rest-auth). Также предоставляет несколько аутентификационных схем и позволяет вам использовать свои схемы.

Система аутификации в Django состоит из:

- Пользователей.

- Разрешений: флаги (yes / no), указывающие, может ли пользователь выполнять определенную задачу.

- Групп. Общий способ применения меток и разрешений для нескольких пользователей.

- Конфигурируемой системы хэширования паролей.

- Форм и средств просмотра для входа в систему пользователей или ограничения содержимого.

Аутентификация всегда запускается в самом начале view, до того как происходить проверка разрешений и ограничений, и до того как начнется выполняться другой код. На рисунках 11 и 12 представлены формы входа и регистрации в приложении.

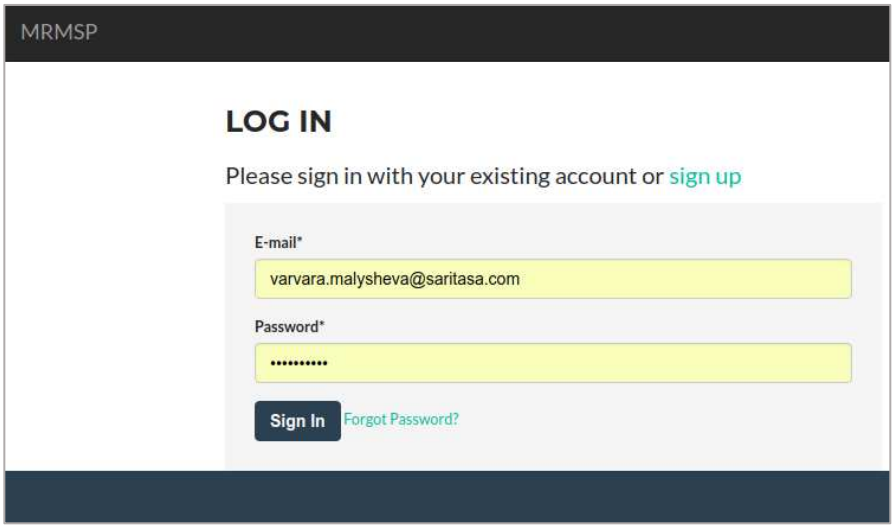


Рисунок 11 - Форма входа в приложение

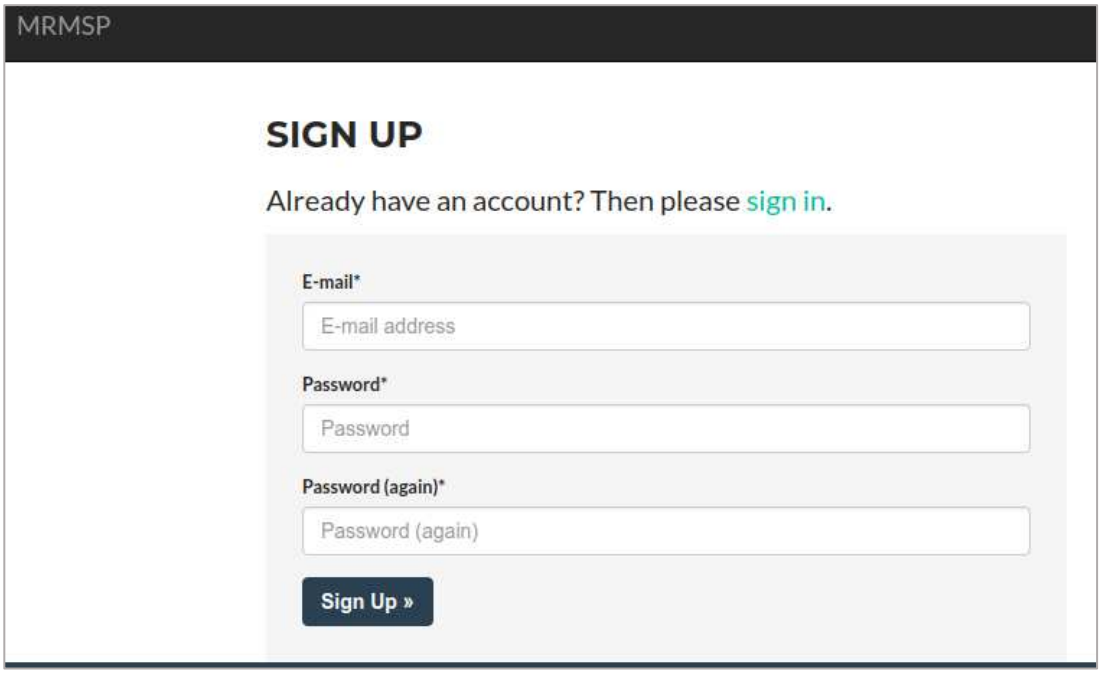


Рисунок 12 - Форма регистрации в приложении

Была сверстана стартовая страница, изображена на рисунке 13. Если пользователь вошел в систему, появляются дополнительные кнопки меню на верхней панели.

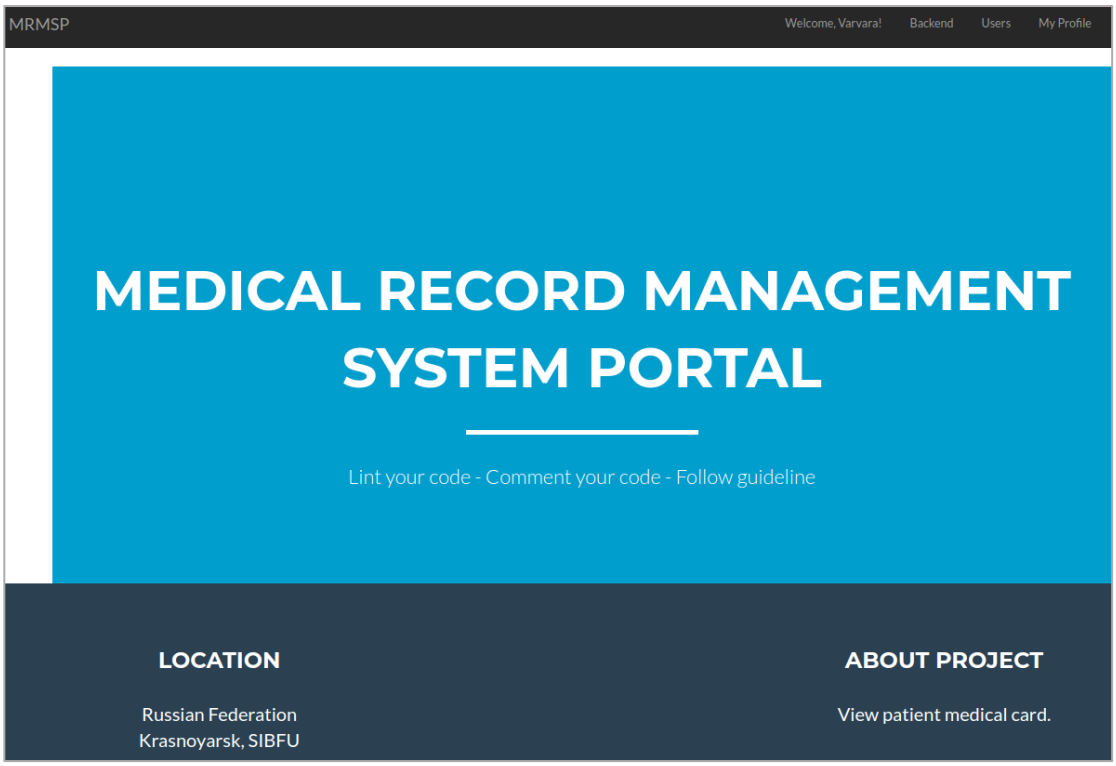


Рисунок 13- Стартовая страница приложения

При регистрации пользователь сначала проходит верификацию email адреса. Затем при первом входе систему заполняются дополнительные данные.

2.4 Реализация просмотра и редактирования пользователя

Одной из самых удобных частей Django является интерфейс автоматического администрирования. Он считывает метаданные из моделей, чтобы обеспечить быстрый, ориентированный на модель интерфейс. Доверенные пользователи могут управлять контентом на сайте.

Сайт администратора включен по умолчанию в каждом сгенерированном проекте Django.

После создания моделей мы теперь готовы добавить их регистрацию с помощью сайта Admin. Для этого нужно определять специальные классы и регистрировать их.

Реализована панель администратора по управлению пользователями. Вид общего списка представлен на рисунке 14.

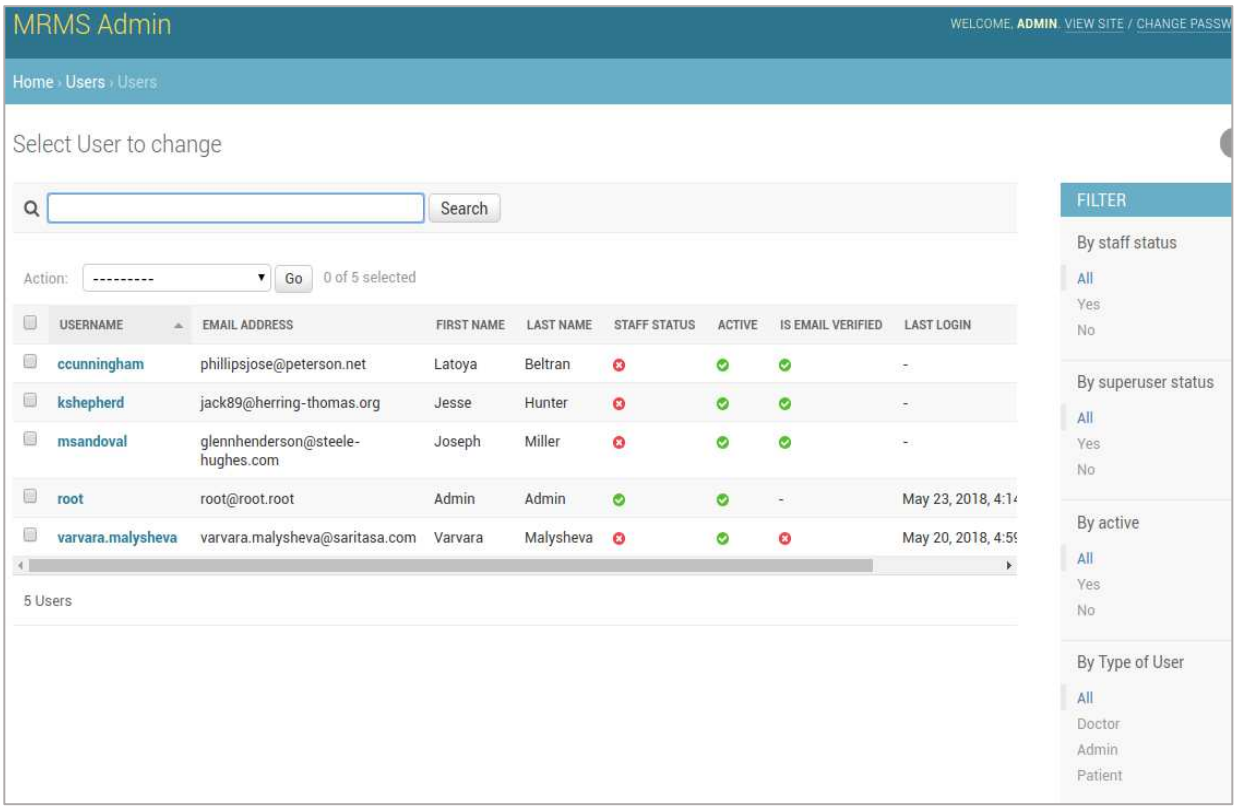


Рисунок 14 - Список пользователей в Django Admin

Страница редактирования пользователя представлена на рисунке 15.

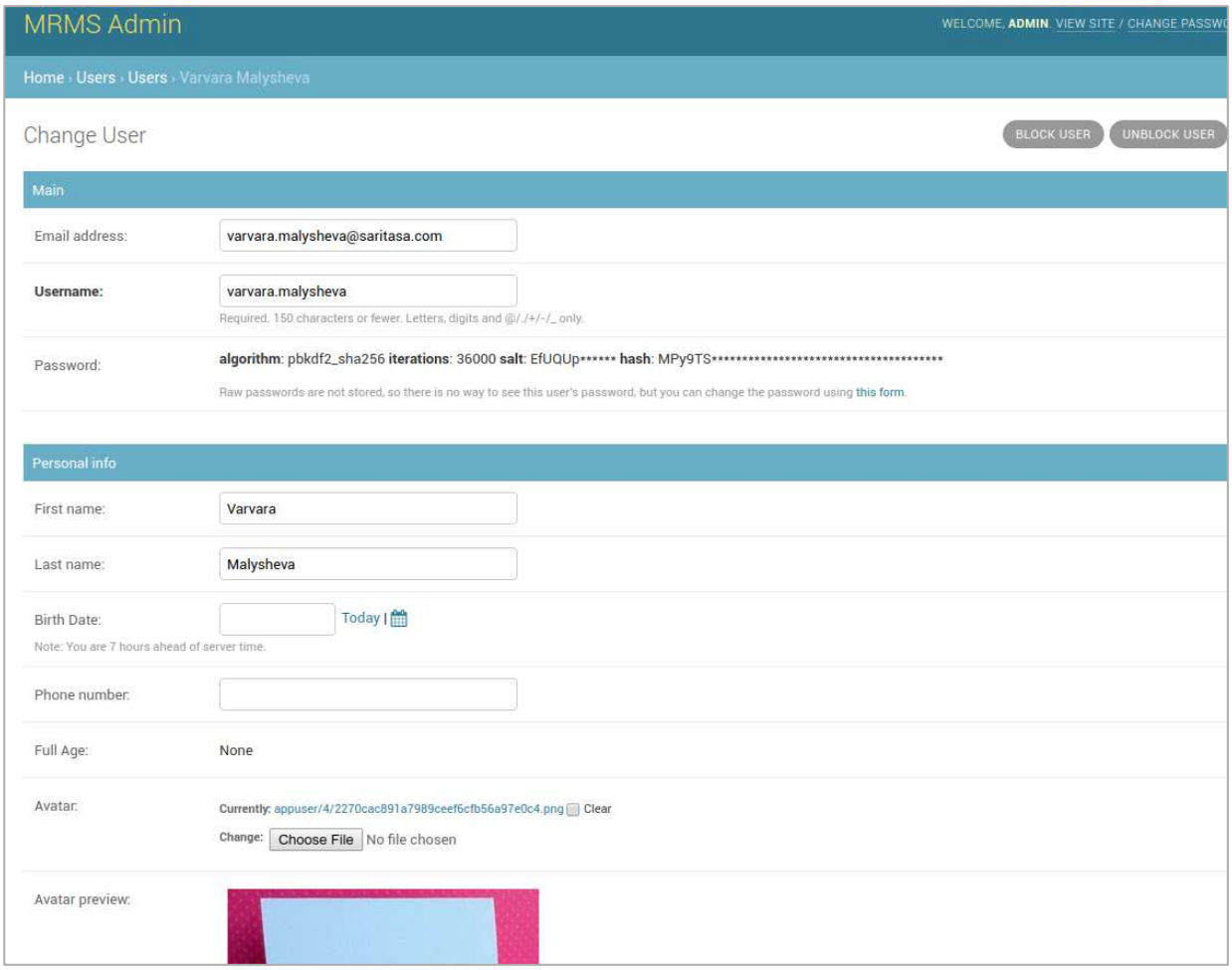


Рисунок 15 - Страница редактирования пользователя

Для реализации это функциональности используется модуль admin в пакете Django. Можно самостоятельно настроить, какие поля отображать, по каким фильтровать и пр.

Также были внедрены две дополнительные кнопки – блокировки и разблокировки пользователя.

Класс ModelAdmin представляет собой представление модели в интерфейсе администратора.

Пользователя может редактировать только администратор не необходимыми правами. Интерфейс представлен на рисунке 16.

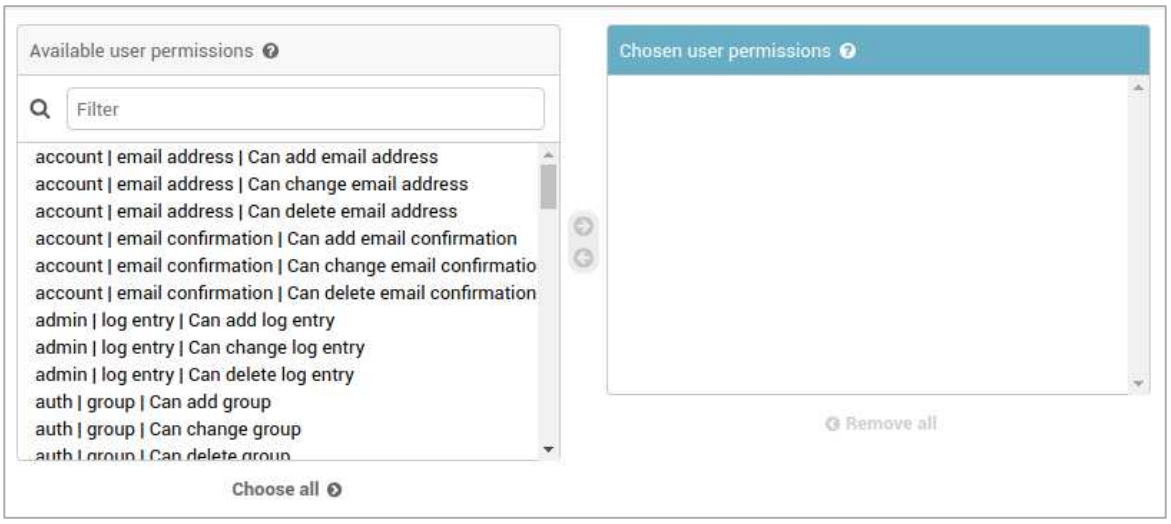


Рисунок 16 - Настройка разрешений пользователя

2.4 Реализация интерфейса медицинской карты

Используется модуль Django tabbed admin – простая библиотека для простого добавления вкладок в админ-формы.

На рисунках 17-21 представлена панель управления медицинской картой пациента.

В каждом блоке можно добавить\удалить сущности (например, анализы и прочее).

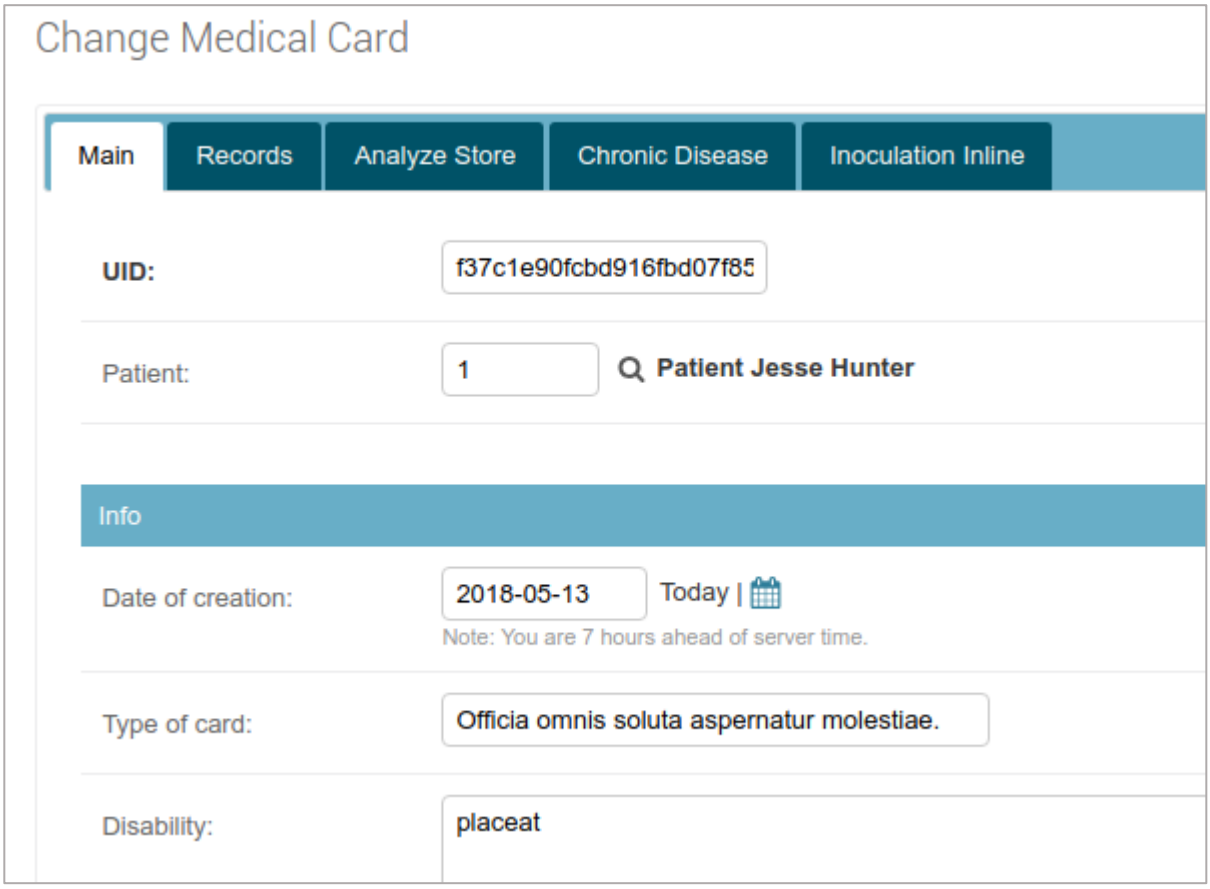


Рисунок 17 - Общий вид экрана редактирования медицинской карты

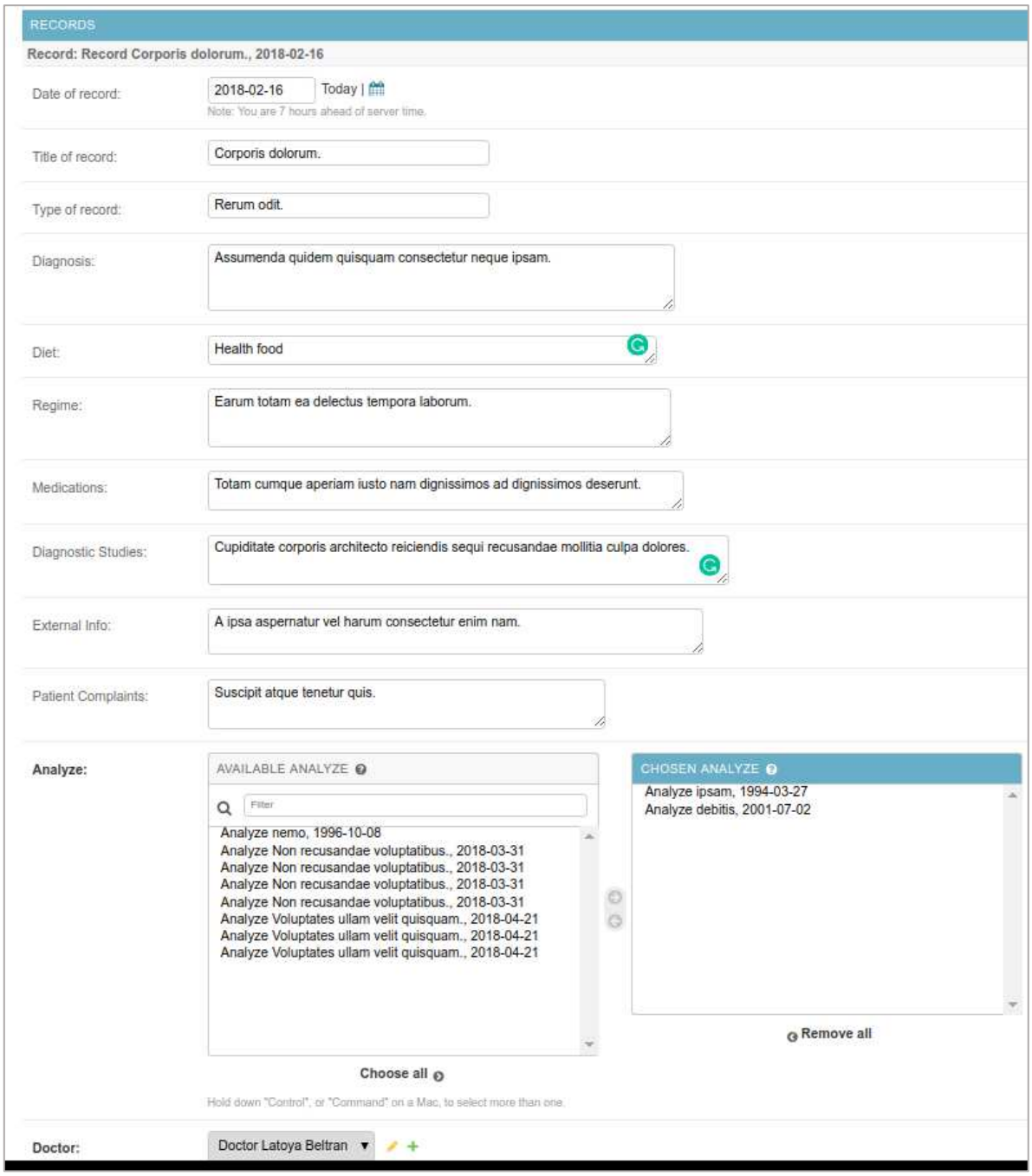


Рисунок 18 - Экран редактирования записей в медицинской карте

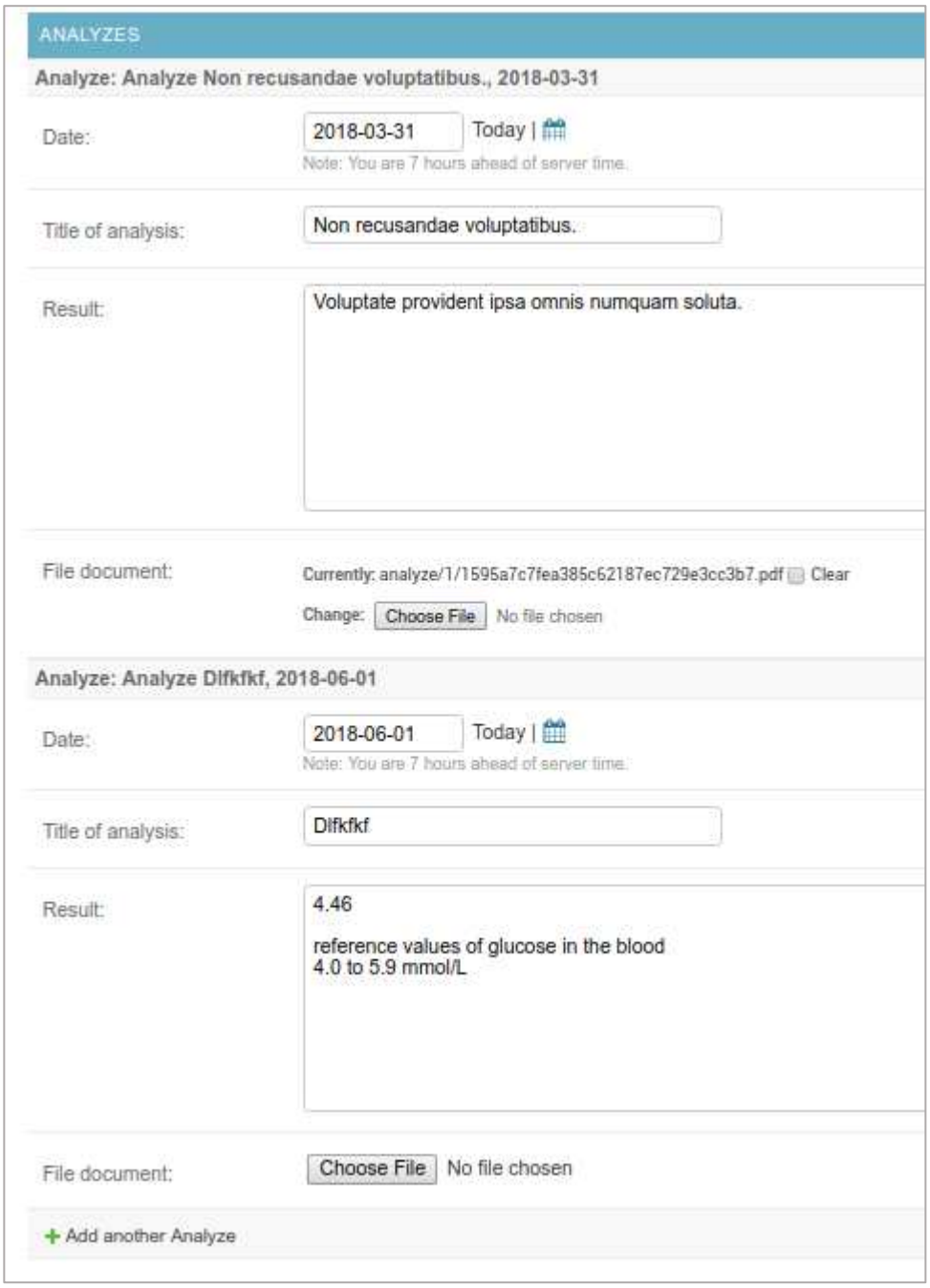


Рисунок 19 - Экран просмотра анализов пациента

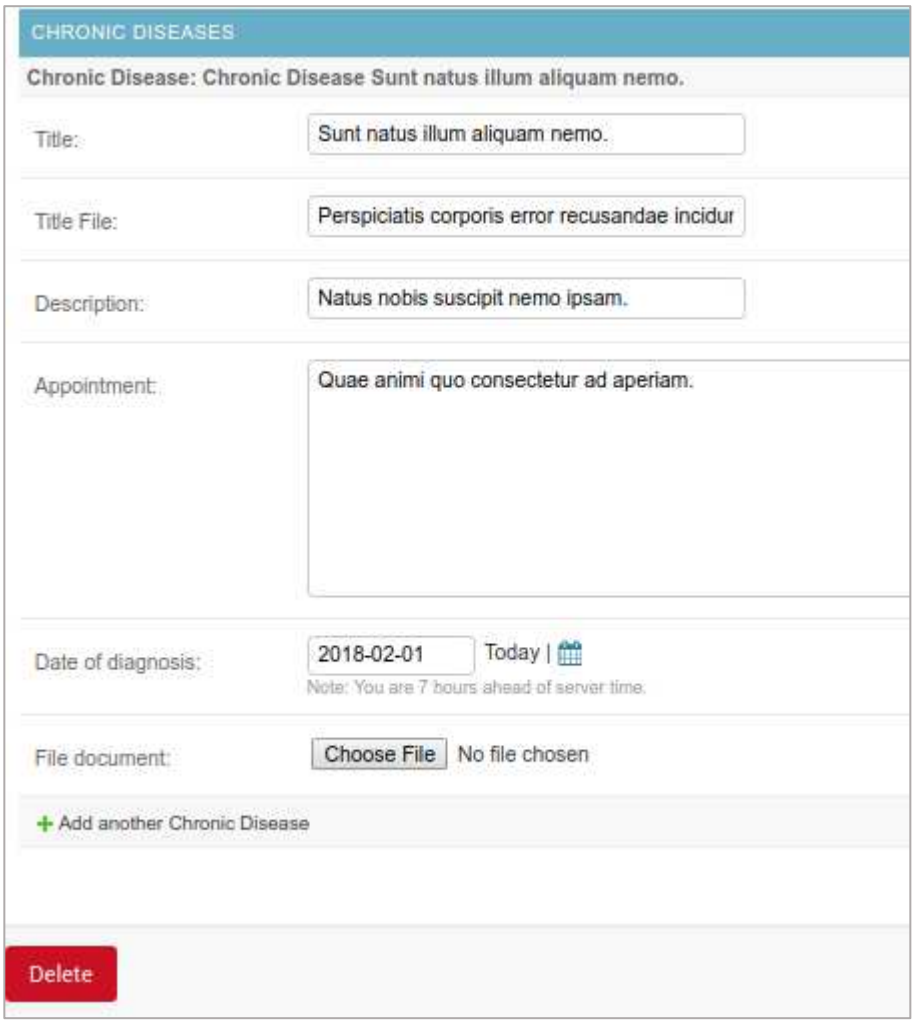


Рисунок 20 - Экран просмотра хронических заболеваний пациента

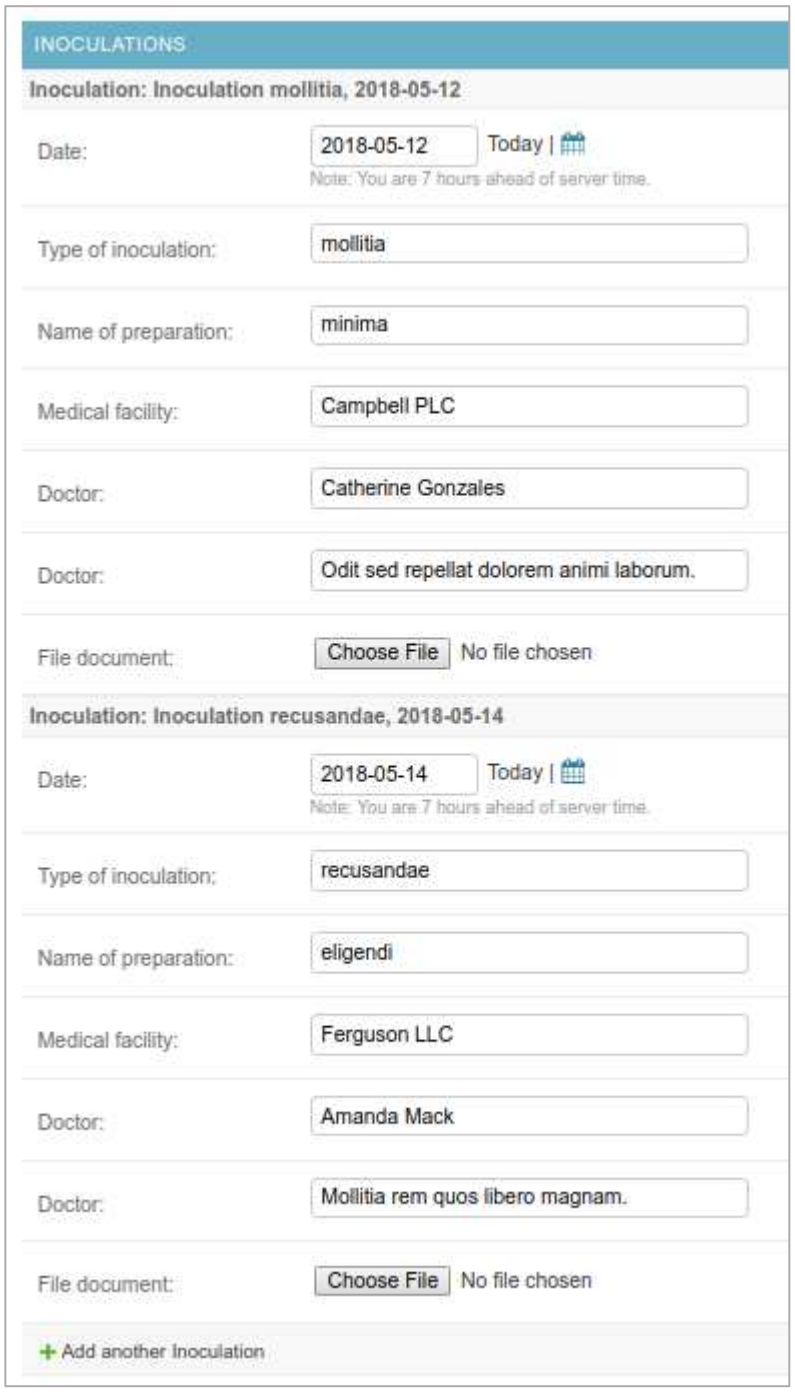


Рисунок 21 - Экран просмотра сделанных прививок пациента

2.5 Реализация просмотра и редактирования данных пользователем

Реализована страница обновления полей пользователя (рисунок 22).

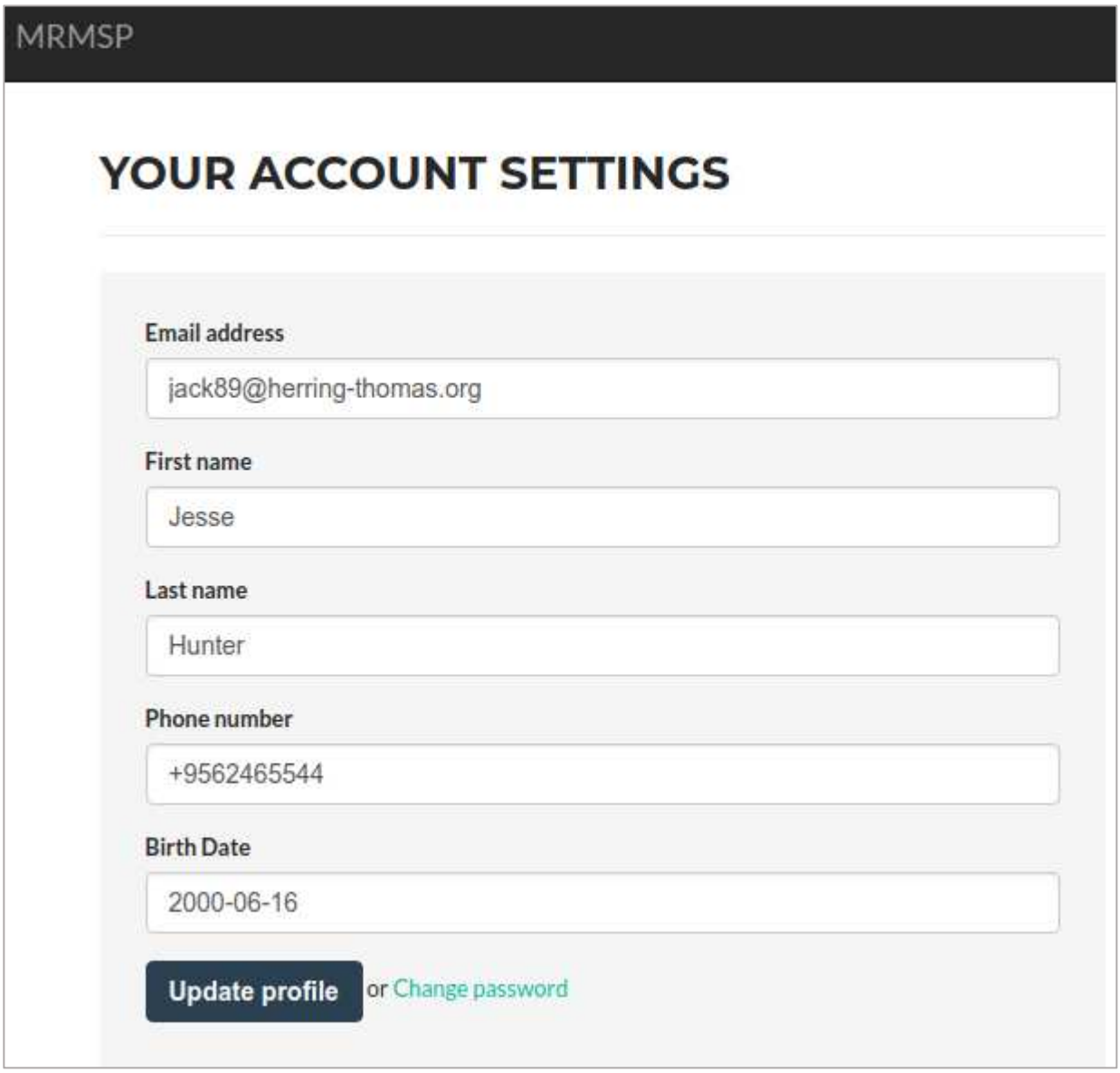


Рисунок 22 - Редактирования профиля

Также была создана форма редактирования данных пользователя как пациента (рисунок 13).

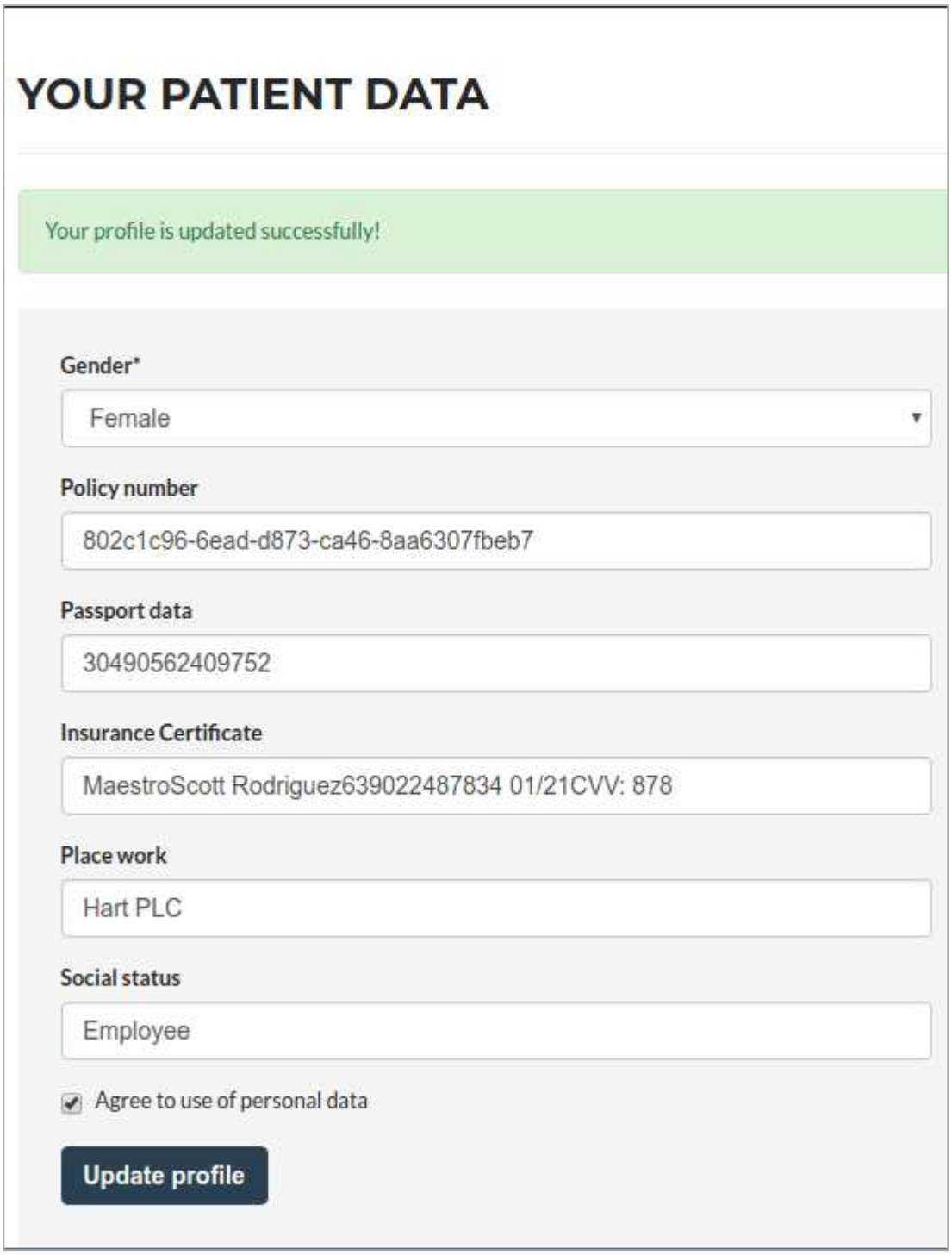


Рисунок 23 - Редактирование данных пациента

Создана таблица расписания врачей (для пациента). Внедрен поиск по всем столбцам (рисунки 24-25).

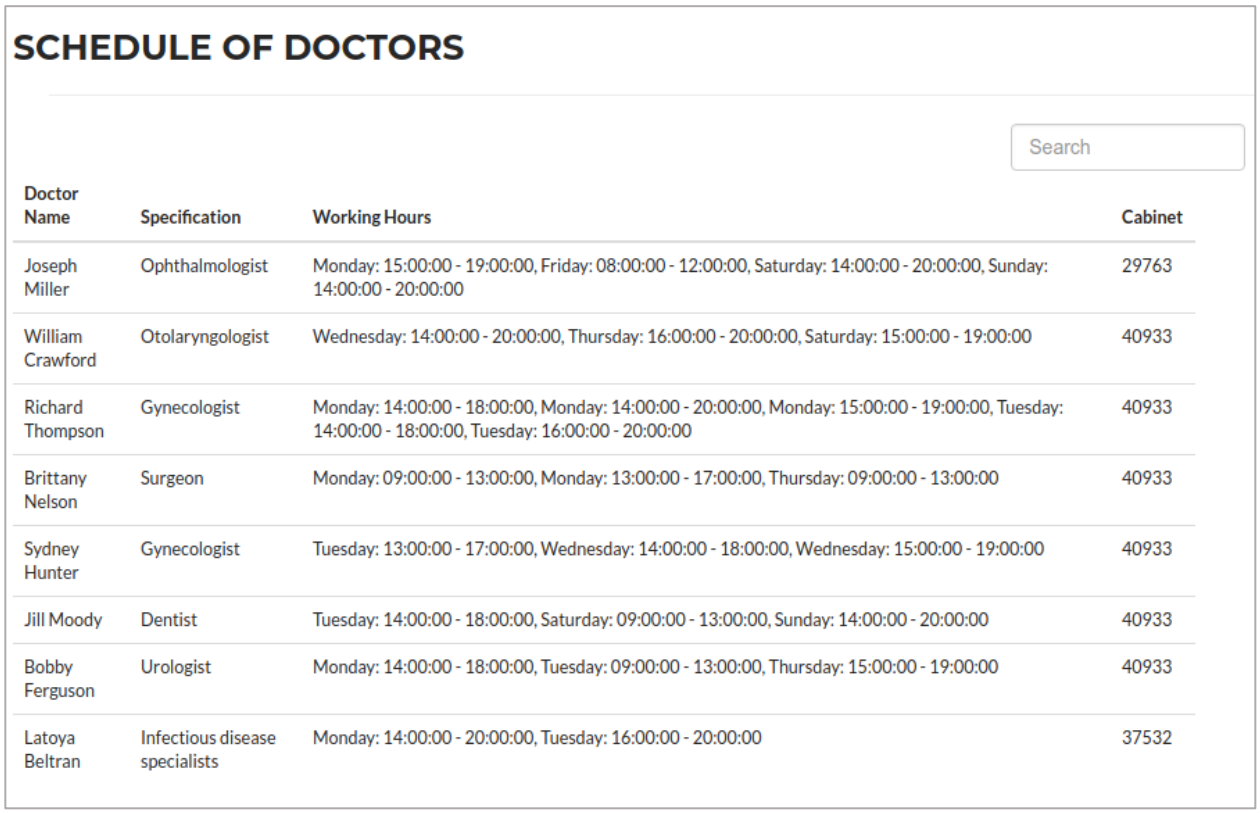


Рисунок 24 – Таблица списка врачей

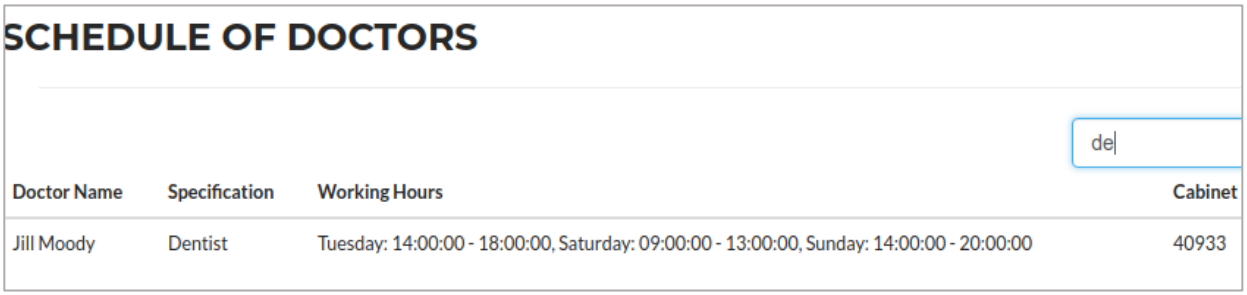


Рисунок 25 – Работа поиска по всем столбцам

Для корректной работы таблиц и поиска было реализована точка API для работы со списком объектов.

Этот интерфейс был реализован с помощью Django REST framework – библиотека предназначена для создания сложных, настраиваемых API и веб сервисов. Были настроены виды, и типы полей, разрешения для JSON ответ (рисунок 26).

Для рендеринга страницы используется загрузочная таблица от фреймворка Boostrap Table (рисунок 27).

```python
from rest_framework import viewsets
from rest_framework import serializers
from rest_framework.permissions import AllowAny

from apps.doctors.models import Doctor


class DoctorSerializer(serializers.ModelSerializer):
    working_hours = serializers.CharField(source='all_working_hours')
    specification_string = serializers.CharField(source='str_specification')
    name_doctor = serializers.CharField(source='name')
    class Meta:
        model = Doctor
        fields = '__all__'


class DoctorViewSet(viewsets.ModelViewSet):
    """
    A simple ViewSet for viewing doctors.
    """
    queryset = Doctor.objects.all()
    serializer_class = DoctorSerializer
    permission_classes = [AllowAny]
```

Рисунок 26 – Реализация списка врачей

```
{% extends "base.html" %}
{% load crispy_forms_tags i18n %}
{% block content %}

    <div class="container">
        <div class="row">
            <h1>Schedule of doctors</h1>
            <hr/>
            <div class="col">
                <table data-toggle="table"
                       data-url="/api/v1/doctors/"
                       data-search-time-out="100"
                       data-id-field="id">
                    <thead>
                    <tr>
                        <th data-field="name_doctor">Doctor Name</th>
                        <th data-field="specification_string">Specification</th>
                        <th data-field="working_hours">Working Hours</th>
                        <th data-field="cabinet">Cabinet</th>
                    </tr>
                    </thead>
                </table>
            </div>
        </div>
    </div>
{% endblock content %}

{% block footer %}
    {% include 'includes/footer.html' with footer_style="navbar-fixed-bottom" %}
{% endblock footer %}
```

Рисунок 27 – Шаблон для отображения таблицы расписания докторов

Django REST framework предоставляет средства для взаимодействия с созданным API. Тестировать и проверять интерфейс можно в браузере (рисунок 28).

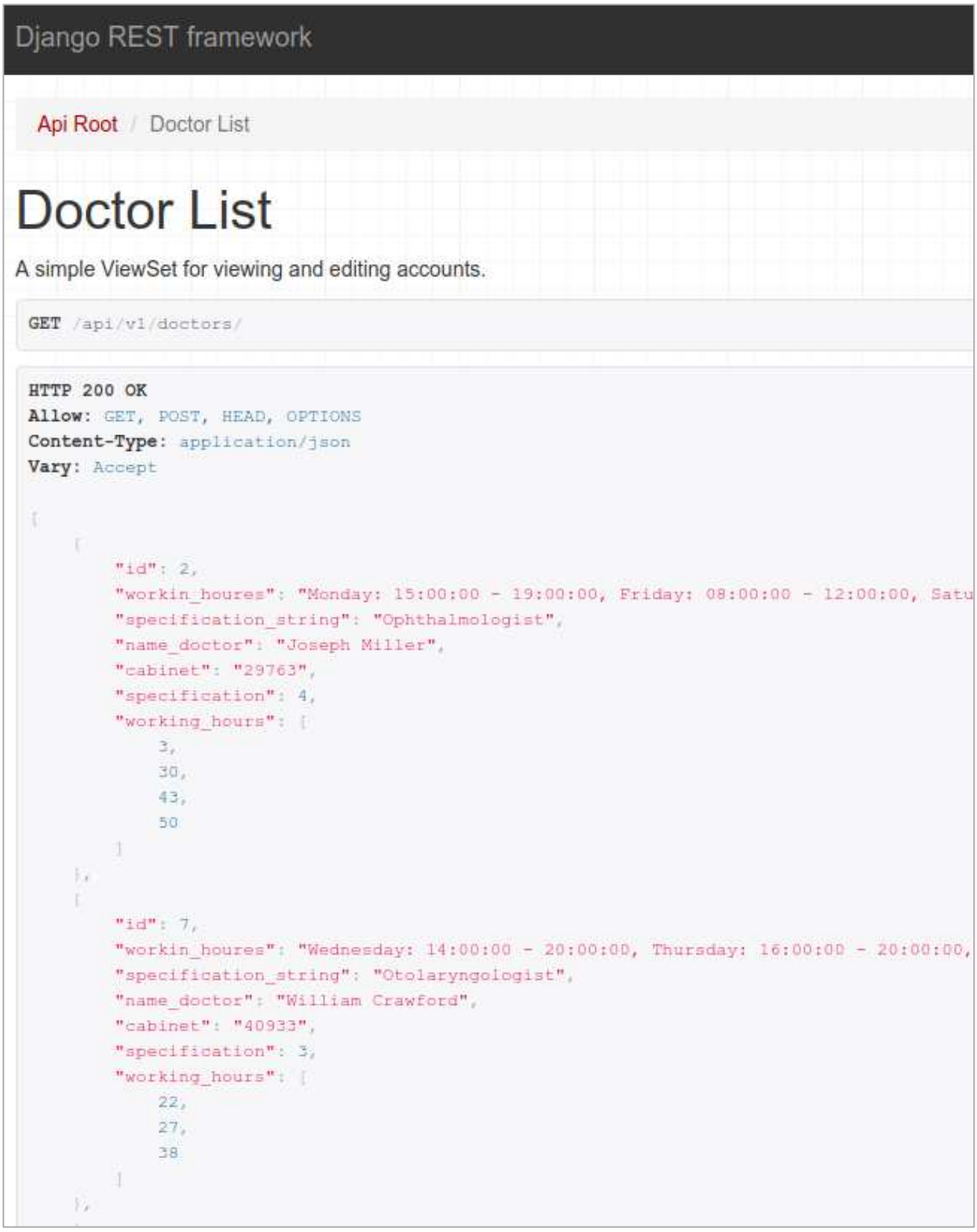


Рисунок 28 – Вид JSON ответа при запросе списка докторов

2.6 Реализация страницы медицинской карты пользователя

Разработан интерфейс просмотра пользователем своих личных данных. Верхнее меню на главной странице показано на рисунке 29.

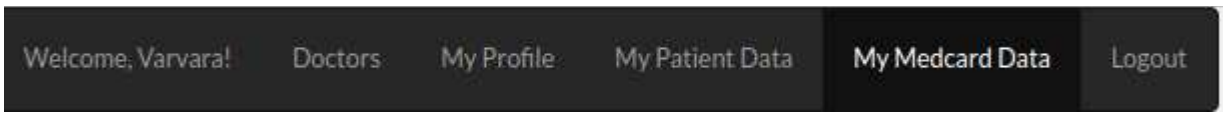


Рисунок 29 – Вид верхнего меню для пользователя Пациент

На рисунках 30 и 31 предствалена реализации страницы медицинской карты пациента. Пользователь может просматривать все свои доступныее данные, проводить поиск. Встроена пагинация по страницам. Также у пользователя есть возможность скачать необходимые документы.

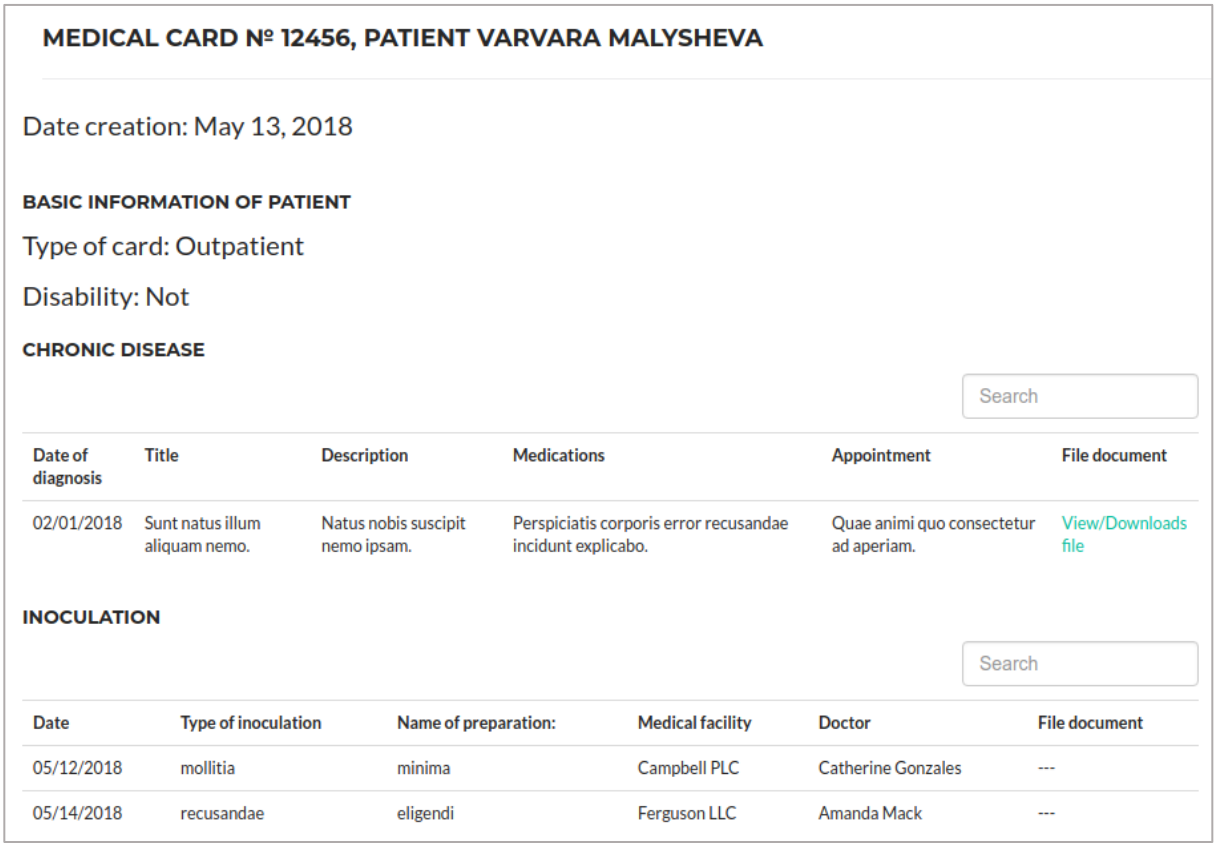
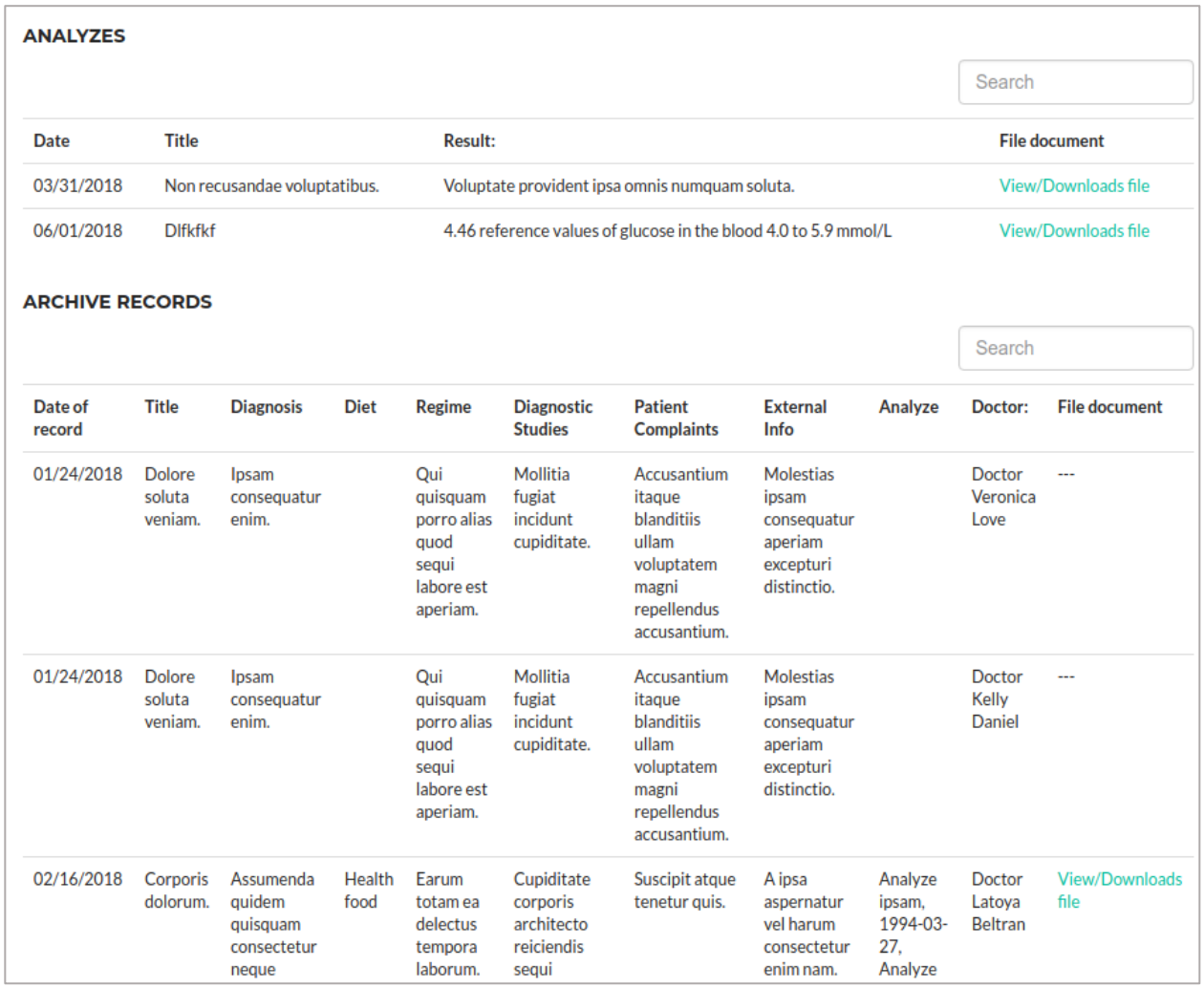


Рисунок 30 Страница просмотра данных медицнской карты



ANALYZES

Search

Date	Title	Result:	File document
03/31/2018	Non recusandae voluptatibus.	Voluptate provident ipsa omnis numquam soluta.	View/Downloads file
06/01/2018	Dlfkfkf	4.46 reference values of glucose in the blood 4.0 to 5.9 mmol/L	View/Downloads file

ARCHIVE RECORDS

Search

Date of record	Title	Diagnosis	Diet	Regime	Diagnostic Studies	Patient Complaints	External Info	Analyze	Doctor:	File document
01/24/2018	Dolore soluta veniam.	Ipsam consequatur enim.		Qui quisquam porro alias quod sequi labore est aperiam.	Mollitia fugiat incidunt cupiditate.	Accusantium itaque blanditiis ullam voluptatem magni repellendus accusantium.	Molestias ipsam consequatur aperiam excepturi distinctio.		Doctor Veronica Love	---
01/24/2018	Dolore soluta veniam.	Ipsam consequatur enim.		Qui quisquam porro alias quod sequi labore est aperiam.	Mollitia fugiat incidunt cupiditate.	Accusantium itaque blanditiis ullam voluptatem magni repellendus accusantium.	Molestias ipsam consequatur aperiam excepturi distinctio.		Doctor Kelly Daniel	---
02/16/2018	Corporis dolorum.	Assumenda quidem quisquam consectetur neque	Health food	Earum totam ea delectus tempora laborum.	Cupiditate corporis architecto reiciendis sequi	Suscipit atque tenetur quis.	A ipsa aspernatur vel harum consectetur enim nam.	Analyze ipsam, 1994-03-27, Analyze	Doctor Latoya Beltran	View/Downloads file

Рисунок 31 Таблицы анализов и записей приема в карте пациента.

ЗАКЛЮЧЕНИЕ

В результате выполнения бакалаврской работы была создана ИС ведения электронных медицинских карт пациентов. В процессе проектирования и разработки были решены все задачи и достигнуты все поставленные цели.

Разработанная в дальнейшем программная система позволит просматривать сводную информацию о врачах (отображение списка врачей с названиями их специальностей) и пациентах (отображение списка пациентов и номеров их карточек), удалять и изменять данные. Предусмотрен интерфейс для пациентов – они смогут просматривать свои личные данные, и записи в своей медицинской карте. Также пациент имеет доступ к расписанию работы докторов.

В ходе выполнения выпускной квалификационной работы были полностью выполнены следующие задачи:

- обоснована актуальность задачи;
- проведен анализ аналогов медицинских систем управления медицинским картами пациентов;
- определены инструментальные средства и начальные требования;
- разработана UML диаграмма классов;
- созданы модели БД посредством использования ORM Django;
- создана БД системы, применены миграции;
- реализован административный интерфейс управления пользователями, пациентами и медицинскими картами в Django Admin;
- реализована стартовая страница приложения;
- внедрена страница просмотра расписания работы врачей;
- внедрено онлайн просмотр пациентами медицинской карты и результатов анализов;
- созданы тестовые данные и проведено тестирование;

Внедрение такой программной системы может значительно сократить время обработки информации, осуществить быстрый поиск необходимой

информации, автоматически формировать отчеты, позволять хранить большие объемы информации и избежать потери информации.

СПИСОК СОКРАЩЕНИЙ

В настоящей работе применены следующие сокращения:

ИС – информационная система;

API – Application Programming Interface (интерфейс для программирования приложений);

БД – база данных;

EMR Electronic Medical Records – (электронные медицинские карты)

ORM – Object-relational mapping, объектно-реляционное отображение;

E/M Coding Evaluation and management coding (кодирование оценки и управления)

HIPAA (Health Insurance Portability and Accountability Act) Закон о переносимости и подотчетности медицинского страхования

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. СТО 4.2-07-2014 Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности. – Введ. 9.01.2014. – Красноярск : ИПК СФУ, 2014. – 60 с.

2. Редмонд, Э. Семь баз данных за семь недель. Введение в современные базы данных и идеологию NoSQL / Э. Редмонд, Д. Р. Уилсон; пер. с англ. Слинкин А. А. – Москва: ДМК Пресс, 2013. – 384с.

3. Бизли Д.М. Python. Подробный справочник, 4-е издание / Д. М. Бизли; пер. с англ. А. Киселёв. Санкт-Петербург: Символ-Плюс, 2010. - 864 с.

4. Головатый, А. Django. Подробное руководство / А. Головатый; пер. с англ. А. Киселев. -, Санкт-Петербург: Символ, 2010. 560 с.

5. Система Electronic Medical Records Software от компании MicroMD. [Электронный ресурс] – Режим доступа – https://www.micromd.com/emr/electronic-medical-record-systems/

6. Система 1С: Медицина. Поликлиника. [Электронный ресурс] – Режим доступа – https://solutions.1c.ru/catalog/clinic/features

7. Хабрахабр - крупнейший ресурс для IT-специалистов. [Электронный ресурс] – Режим доступа – https://habrahabr.ru/

8. Python - официальная документация языка [Электронный ресурс] – Режим доступа – https://www.python.org/

9. Python Cookbook by David Beazley, Brian K. Jones (russian translate / перевод на русский язык). [Электронный ресурс]. – Режим доступа – https://github.com/borisuvarov/python-cookbook-ru

10. Официальная документация Django. [Электронный ресурс] – Режим доступа – https://www.djangoproject.com

11. Свободная энциклопедия Википедия [Электронный ресурс] Django – Декабрь 2017, 24. – Режим доступа: https://ru.wikipedia.org/wiki/Django

12. Bootstrap – официальная документация [Электронный ресурс] / режим доступа: https://getbootstrap.com/docs/3.3/getting-started/

13. Библиотека django-rest-auth – официальная документация [Электронный ресурс] / режим доступа: https://django-rest-auth.readthedocs.io/en/latest/

14. PostgresSQL– официальная документация [Электронный ресурс] / режим доступа: https://www.postgresql.org/docs/9.0/static/sql-do.html

15. Git – официальная документация [Электронный ресурс] / режим доступа: https://git-scm.com/

16. Docker – официальная документация [Электронный ресурс] / режим доступа: https://docs.docker.com/

17. Библиотека для создания тестовых данных Factory-boy– официальная документация [Электронный ресурс] / режим доступа: http://factoryboy.readthedocs.io/en/latest/index.html

18. Django Rest framework – библиотека расширенного инструментария API – официальная документация [Электронный ресурс] / режим доступа: http://www.django-rest-framework.org/#api-guide

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и Информационных технологий
институт
Информационные системы
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой ИС
_____ Л.С. Троценко
подпись инициалы, фамилия
« 13 » июня 2018 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 Информационные системы и технологии

Информационная система ведения медицинских карт пациентов

Руководитель _____ 13.06.18 И.А. Легалов
подпись, дата инициалы, фамилия

Выпускник _____ 13.06.18 В.А. Малышева
подпись, дата инициалы, фамилия

Нормоконтролер _____ 13.06.18 Ю.В. Шмагрис
подпись, дата инициалы, фамилия

Красноярск 2018