



Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

«Разработка веб-приложения для контроля процесса приема
пациента у врача в медицинской клинике»

Студент ИУ7-66Б
(Группа)

(Подпись, дата)

Ж.Р.Турсунов
(И.О.Фамилия)

Руководитель курсового проекта

(Подпись, дата)

Ю.М.Гаврилова
(И.О.Фамилия)

Консультант

(Подпись, дата)

(И.О.Фамилия)

Москва, 2021 г.

Министерство науки и высшего образования Российской Федерации Федеральное
государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э.Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э.Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой ИУ7
(Индекс)

_____ И.В.Рудаков
(И.О.Фамилия)

« ____ » _____ 2021 г.

ЗАДАНИЕ

на выполнение курсовой работы

по дисциплине Базы Данных

Студент группы ИУ7-66Б

_____ Турсунов Жасурбек Рустамович

(Фамилия, имя, отчество)

Тема курсовой работы Разработка веб-приложения для контроля процесса приема пациента
у врача в медицинской клинике

Направленность КР (учебная, исследовательская, практическая, производственная, др.)

_____ учебная

Источник тематики (кафедра, предприятие, НИР) _____ кафедра

График выполнения проекта: 25% к ____ нед., 50% к ____ нед., 75% к ____ нед., 100% к ____ нед.

Задание

Необходимо создать БД для хранения историй болезней пациентов. Каждый пациент может записаться на несколько приемов. В приложении должен быть необходимый функционал для загрузки необходимых медицинских файлов (МРТ, ЭЭГ, КТ и т.д). Требуется также реализовать пункты для выбора услуг, которыми воспользовался пациент во время приема. Исходя из этих выборов, в итоге должна сформулироваться итоговая сумма для оплаты. После отметки пункта, об успешной оплате оказанных услуг, в приложении должна активироваться кнопка «Скачать», для загрузки выписки на лечение для дальнейшей её печати.

Оформление курсовой работы:

Расчетно-пояснительная записка на ____ листах формата А4.

Расчетно-пояснительная записка должна содержать постановку введение, аналитическую часть, конструкторскую часть, технологическую часть, экспериментально-исследовательский раздел, заключение, список литературы, приложения.

Дата выдачи задания « ____ » _____ 2021 г.

Руководитель курсового проекта

_____ (Подпись, дата)

Ю.М.Гаврилова
(И.О.Фамилия)

Студент

_____ (Подпись, дата)

Ж.Р.Турсунов
(И.О.Фамилия)

Содержание

Введение	4
1 Аналитическая часть	5
1.1 Обзор и анализ существующего программного обеспечения и обоснование необходимости разработки	5
1.1.1 Medesk	5
1.1.2 MEDODS	6
1.1.3 Инфоклиника	7
1.1.4 Вывод	8
1.2 Типы и выбор Баз Данных	8
1.2.1 Иерархическая база данных	8
1.2.2 Сетевые базы данных	9
1.2.3 Реляционные базы данных	10
1.2.4 Вывод	11
2 Конструкторская часть	12
2.1 Разработка структуры БД	12
2.2 Use-Case диаграмма БД с выделением акторов	13
2.3 IDEF0 диаграмма для функции авторизации	13
2.4 IDEF0 диаграмма для функции скачивания рецепта на лечение	14
2.5 Вывод из конструкторской части	14
3 Технологическая часть	15
3.1 Архитектура программного обеспечения	15
3.2 Выбор среды и языка разработки серверной части	15
3.3 Используемые инструменты и технологии веб-приложения	16
3.4 Реализация	16
3.5 Интерфейс программы	16
3.6 Листинг кода	20
3.7 Вывод из технологической части	21
4 Исследовательская часть	22
4.1 Системные характеристики	22
4.2 Постановка эксперимента	22
4.3 Вывод из исследовательской части	22
Заключение	23
Список литературы	24

Введение

Роль компьютеров с каждым днём становится всё больше и больше. С каждым разом увеличивается область применения этой технологии. Для каждой области применения необходимы свои собственные программы, для решения конкретных задач. Благодаря этому в настоящее время постоянно появляются новые предметы изучения и исследования. Область медицины не стало исключением. Современные компьютерные разработки оказывают положительное влияние на развитие новых способов организации медицинской помощи населению. Большое количество стран уже давно активно используют новые технологии в сфере здравоохранения. Проведение телеконсультаций пациентов и персонала, обмен информацией о больных между различными учреждениями, дистанционное фиксирование физиологических параметров, контроль за проведением операций в реальном времени — все эти возможности дает внедрение информационных технологии в медицину. Это выводит информатизацию здравоохранения на новый уровень развития, положительно сказываясь на всех аспектах его деятельности.

Внедрение компьютерных технологий в сферу здравоохранения позволяет улучшить качество обслуживания, заметно ускорить работу персонала и снизить затраты на обслуживание для пациентов. Эти преимущества теперь доступны каждой клинике. Современное программное обеспечение MedLight позволит такую возможность каждому своему пользователю. Программа, обеспечит эффективное хранение и обработку огромных массивов медицинских данных, тем самым убрав из оборота все бумажную работу. Администрации не придется собирать данные вручую для формирования отчета, бухгалтерам в свою очередь будет легче контролировать денежные поступления - все это будет доступно в программе. Эта система, позволит вывести учреждение на новый уровень обслуживания и работы.

Цель данной работы - решить проблемы предметной области, такие как:

1. Проблема надёжности хранения данных;
2. Использование большого объема бумажных документов;
3. Трата большого количества времени при написании однотипных лечений;
4. Жесткий контроль оплаты всех финансовых операций;

1 Аналитическая часть

1.1 Обзор и анализ существующего программного обеспечения и обоснование необходимости разработки

На сегодняшний день существуют множество программ, направленные вести контроль прием пациентов в медицинском учреждении. К более известным можно отнести следующие программы:

1. Medesk;
2. MEDODS;
3. Инфоклиника;
4. МедАнгел.

1.1.1 Medesk

Medesk — медицинская платформа для эффективного управления клиникой. Функционал включает онлайн-запись, работу с протоколами и лабораториями в одном окне, электронные медицинские карты, онлайн-расписание врачей, автоматические напоминания, склад, аналитику и отчетность.

Недостатки:

- очень дорогая стоимость ПО;
- нет возможности отказаться от некоторого функционала.

Преимущества:

- очень много полезных интеграций с многими организациями.

На Рисунке 1 показан внутренний интерфейс программы Medesk.

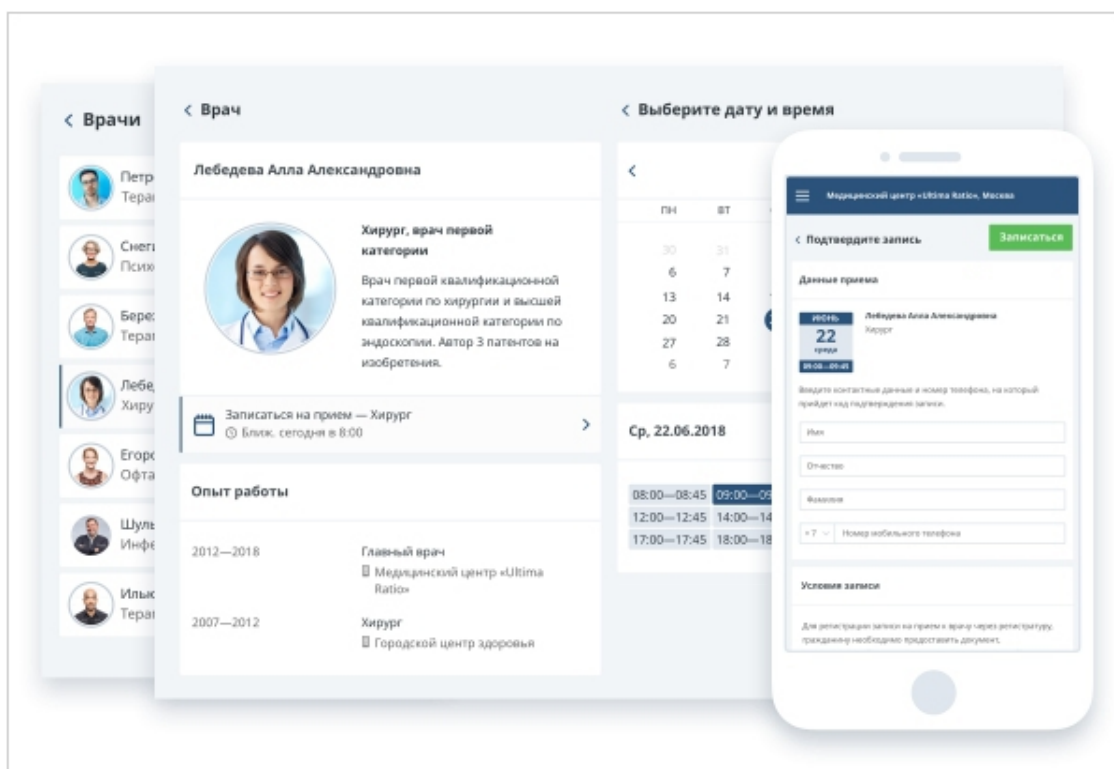


Рис. 1: Интерфейс программы Medesk.

1.1.2 MEDODS

MEDODS - платформа для организации работы частной медицинской и стоматологической клиники. Позволяет эффективно организовать работу клиники: записывать пациентов на прием, вести электронные медицинские карты, выставлять счета, автоматически формировать договоры, получать сводную статистику работы и многое другое.

Недостатки:

- сложный интерфейс;
- нет возможности вносить свои изменения.

Преимущества:

- возможность совершать телефонные вызовы прямо в приложении.

На Рисунке 2 показан внутренний интерфейс программы MEDODS.

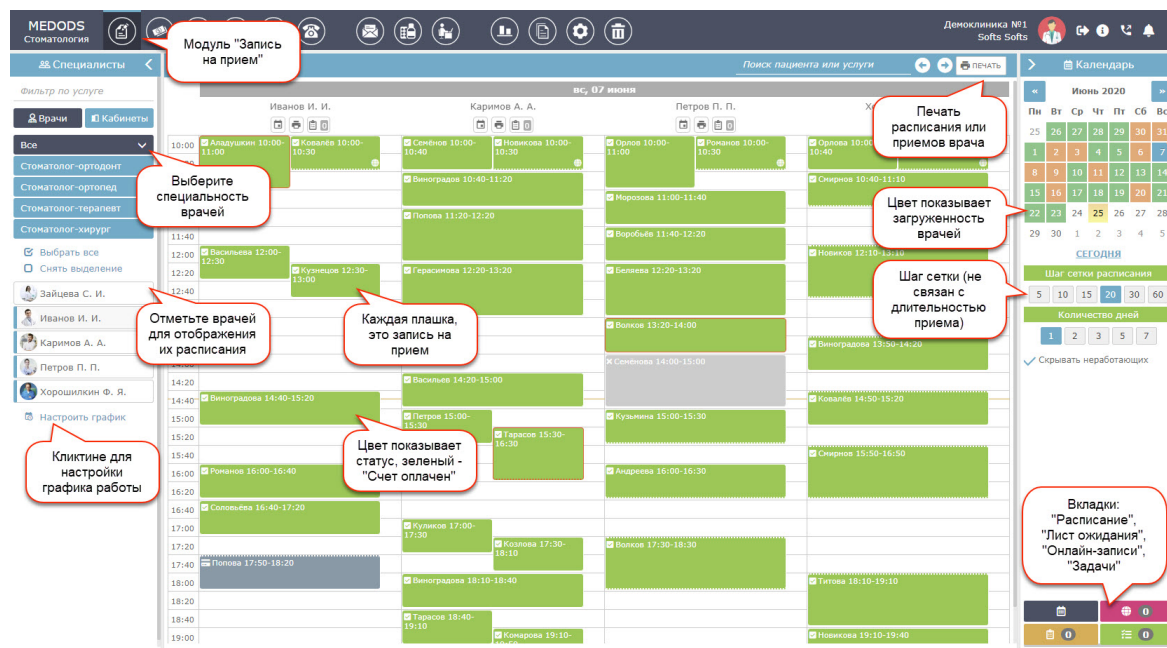


Рис. 2: Интерфейс программы MEDODS.

1.1.3 Инфоклиника

Инфоклиника - полнофункциональная медицинская информационная система: управление поликлиникой, больницей, медицинским центром и сетью медицинских учреждений + SaaS решение для организации сайта электронной регистратуры и личного кабинета пациента клиники.

Недостатки:

- давно не обновлялся;
- очень сложный интерфейс.

Преимущества:

- возможность проводить обучения для пользователей;
- бесплатный пробный период.

На Рисунке 3 показан внутренний интерфейс программы Инфоклиника.

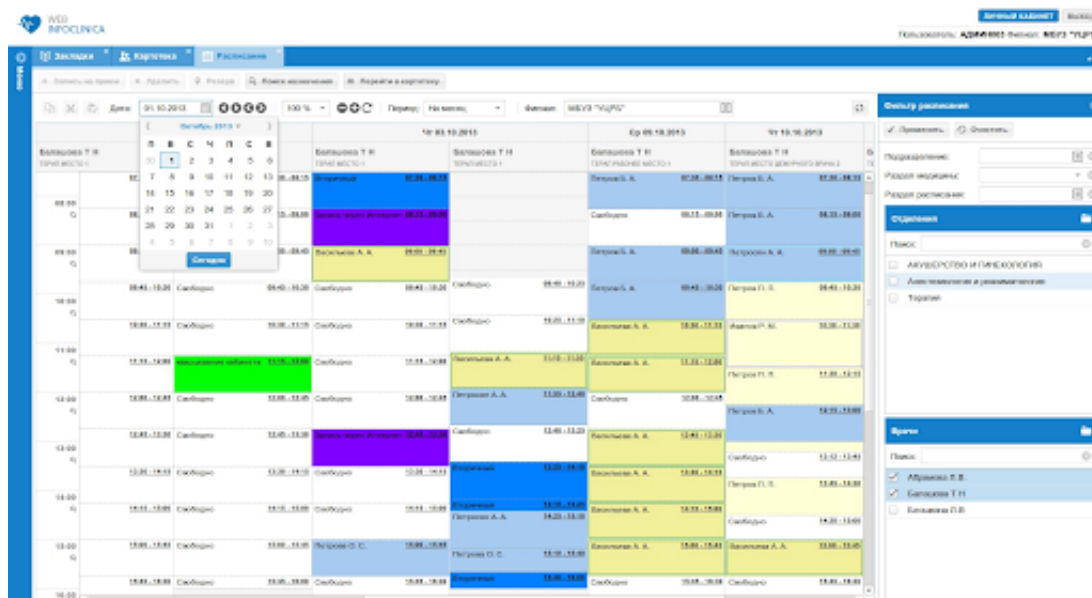


Рис. 3: Интерфейс программы Инфоклиника.

1.1.4 Вывод

Не смотря на то, что существует более чем достаточное количество программного обеспечения для ведения учета в медицинском учреждении, характерными минусами для всего программного обеспечения является высокие требования к аппаратным ресурсам и высокая цена. В своей программе я устраню эти недостатки, сделав его более гибким и доступным для пользователя.

1.2 Типы и выбор Баз Данных

Типы баз данных, называемых также моделями БД или семействами БД, представляют собой шаблоны и структуры, используемые для организации данных в системе управления базами данных (СУБД). Выбор типа повлияет на то, какие операции сможет выполнять приложение, как будут представлены данные, на функции СУБД для разработки и рантайма. Разделяют следующие основные виды БД:

1. иерархическая база данных;
2. сетевые базы данных;
3. реляционные базы данных.

1.2.1 Иерархическая база данных

Иерархическая база данных - каждый объект при таком хранении информации представляется в виде определенной сущности, то есть, у этой сущности могут быть дочерние элементы, родительские элементы, а у тех дочерних могут быть еще дочерние элементы, но есть один объект, с которого все начинается. Получается своеобразное дерево. Примером иерархической базы данных может быть, документ в формате XML или файловая система компьютера.

Следует сказать, что базы данных подобного вида оптимизированы под чтение информации, то есть, базы данных, имеющие иерархическую структуру умеют очень быстро выбирать, запрашиваемую информацию и отдавать ее пользователям. Но такая структура не позволяет столь же быстро перебирать информацию, тут можно привести пример из жизни, компьютер может легко работать с каким-либо конкретным файлом или папкой (которые, по сути являются объектами иерархической структуры) но проверка компьютера антивирусам осуществляется очень долго. Вторым примером – реестр Windows.

На Рисунке 4 можно увидеть структуру иерархической базы данных, в самом верху находится родитель или корневой элемент, ниже находятся дочерние элементы, элементы, находящиеся на одном уровне называются братьями, ну или соседними элементами. Соответственно чем ниже уровень элемента, тем вложенность этого элемента больше.

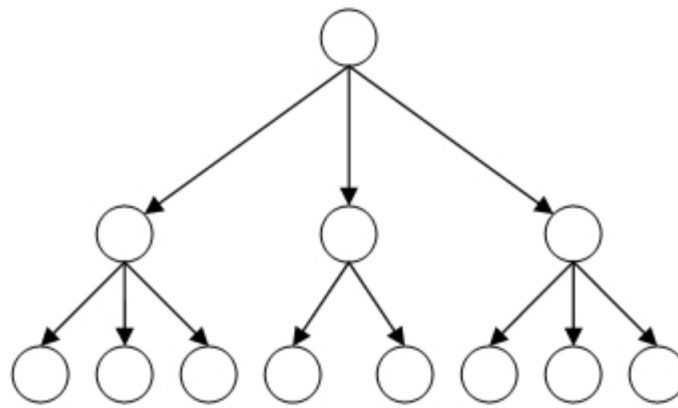


Рис. 4: Структура иерархической базы данных.

1.2.2 Сетевые базы данных

Сетевые базы данных - являются своеобразной модификацией иерархических баз данных. Если обратить внимание на Рисунок 4, можно заметить, что к каждому нижнему элементу идет только одна стрелочка от верхнего элемента. То есть у иерархических баз данных у каждого дочернего элемента может быть только один потомок. Сетевые базы данных отличаются от иерархических тем, что у дочернего элемента может быть несколько предков, то есть, элементов стоящих выше него. Для большей наглядности и понимания структуры сетевых баз можно увидеть на Рисунке 5.



Рис. 6: Структура реляционной базы данных.

1.2.4 Вывод

Не смотря на то, что существует разные типы баз данных, в данной работе была выбрана реляционная модель. Эта модель была выбрана за следующие достоинства:

1. простота и доступность для понимания пользователем. Единственной используемой информационной конструкцией является "таблица";
2. строгие правила проектирования, базирующиеся на математическом аппарате;
3. полная независимость данных. Изменения в прикладной программе при изменении реляционной БД минимальны;
4. для организации запросов и написания прикладного ПО нет необходимости знать конкретную организацию БД во внешней памяти.

2 Конструкторская часть

2.1 Разработка структуры БД

На Рисунке 7 показана ER-диаграмма, описывающая концептуальные схемы предметной области.

База данных состоит из 5 таблиц:

1. patients;
2. records;
3. doctors;
4. services;
5. discharge.

В таблице Patients хранится уникальная информация о пациенте, а именно его имя, дата рождения и номер телефона. Каждый пациент может иметь неограниченное число посещений у врача, в следствие чего, создана таблица Records, которая направлена на хранение информации о каждом приеме у врача. Эта модель тесно связана с таблицами Doctors, Services, Discharge.

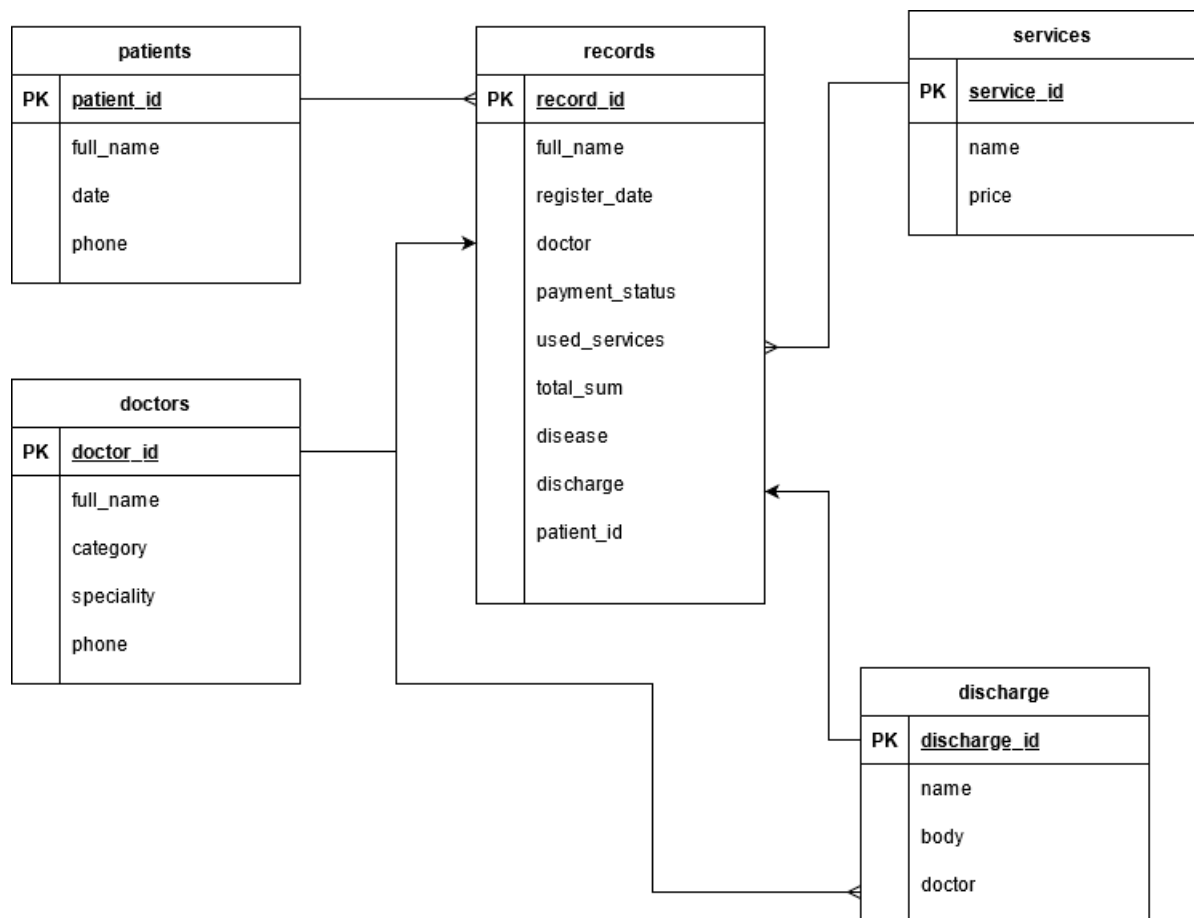


Рис. 7: ER-модель структуры БД

2.2 Use-Case диаграмма БД с выделением акторов

На Рисунке 8 показана Use-Case диаграмма БД с выделением акторов, описывающая структуру и логику данного ПО. На рисунке видно, как выделено 3 актора: Регистратура, Врач, Пациент.

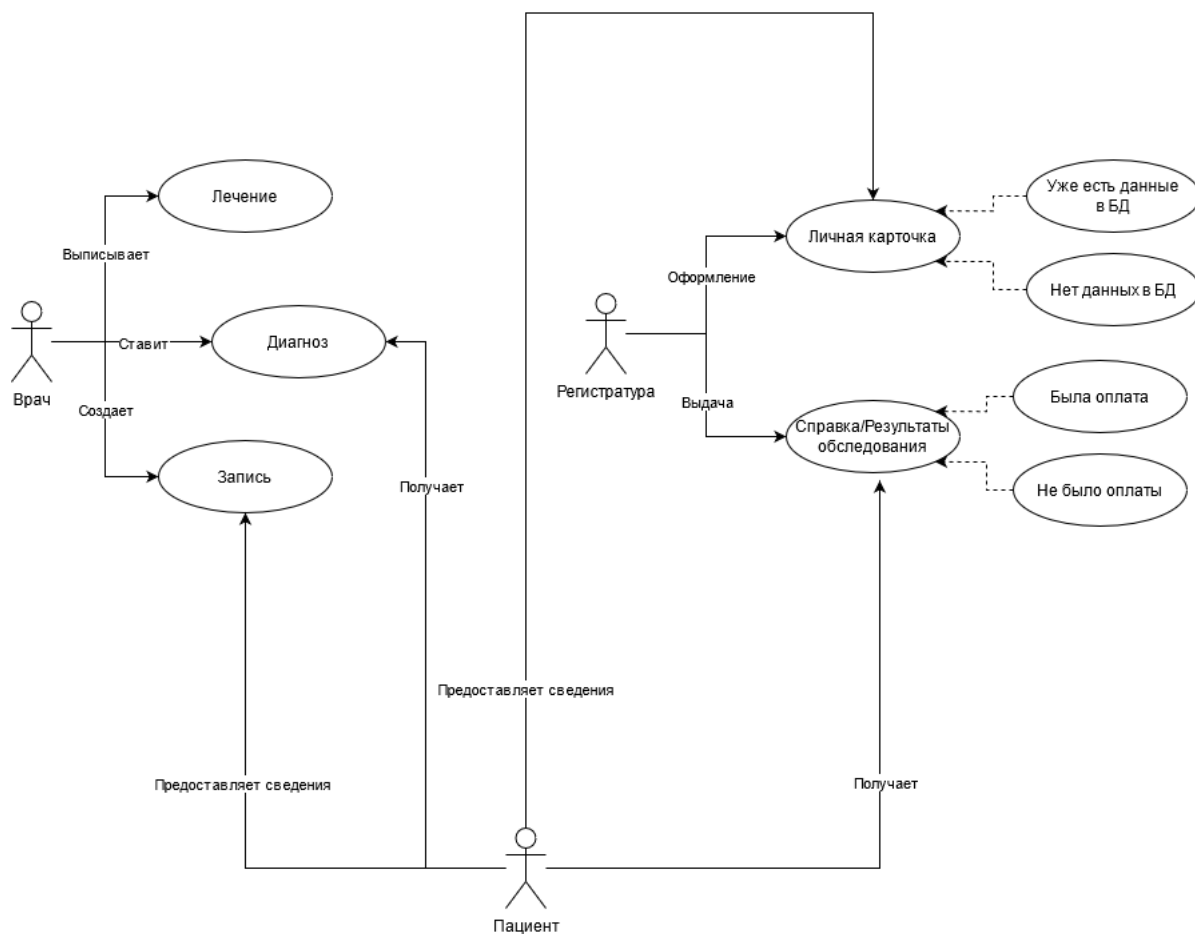


Рис. 8: Use-Case диаграмма БД с выделением акторов

2.3 IDEF0 диаграмма для функции авторизации

На Рисунке 9 показана IDEF0 диаграмма Функции авторизации.

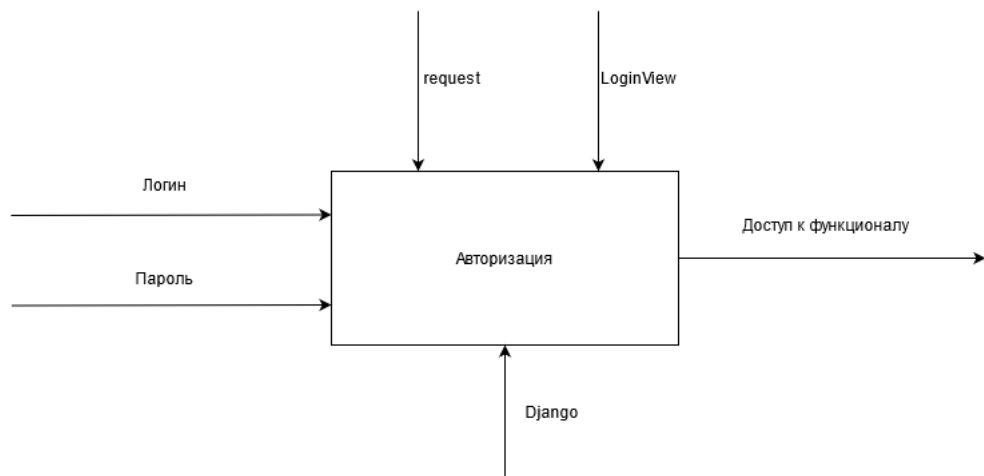


Рис. 9: IDEF0 диаграмма для функции авторизации

2.4 IDEF0 диаграмма для функции скачивания рецепта на лечение

На Рисунке 10 показана IDEF0 диаграмма для функции скачивания рецепта на лечение. Стоит отметить что данная функция доступна только тогда, когда пациент произведет оплату за оказанные ему услуги. Данный статус может быть отмечен только регистратором или кассиром. У врачей данная функция отсутствует, ввиду того, чтобы решить проблему с финансовостью.

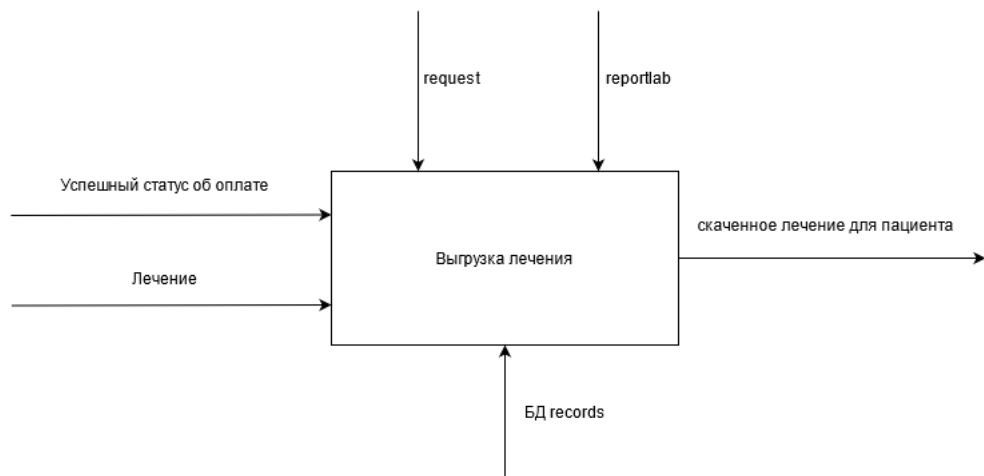


Рис. 10: IDEF0 диаграмма для функции скачивания рецепта на лечение

2.5 Вывод из конструкторской части

В данном разделе были представлены схемы и диаграммы представляющую структуру БД, а также описывающую логику ее работы. Так же были показаны диаграммы некоторых основных функций ПО.

3 Технологическая часть

В данном разделе будут рассмотрены требования к программному обеспечению, средства реализации.

3.1 Архитектура программного обеспечения

В соответствии с техническим заданием будущее приложение должно иметь трехзвенную систему: веб-приложение(представление), базу данных(модель) и логику на серверной части(контроллер). Основная цель применения этой концепции состоит в отделении бизнес-логики(модели) от её визуализации. За счёт такого разделения повышается возможность повторного использования кода.

3.2 Выбор среды и языка разработки серверной части

При выборе языка будем руководствоваться следующими факторами:

1. платформ независимый;
2. динамический;
3. высокоуровневый;
4. многопоточный.

Под динамическим в данном случае понимается, что программа может выполнять обширное количество задач во время обработки информации, которая может быть использовано для проверки и разрешения доступа к объектам на время выполнения.

Всем этим критериям удовлетворяет язык Python3. Основным достоинством является то, что с помощью данного языка можно «разбить» реализацию: интерфейс, работу с файлами и элементами формы организовать с помощью встроенных средств.

При выборе среды разработки рассматриваются и другие возможные среды, которые помогут автоматизировать процесс разработки. Для разработки данной программы необходимо обеспечение средой следующих возможностей:

1. удобные инструменты для отладки и поиска ошибок, в случае их возникновения;
2. разработка более гибкой и надежной программы путем обработки различных исключительных ситуаций, возникающих в результате некорректной работы программы;
3. использование всплывающих подсказок во время написания кода программы, что обеспечивает значительное экономии времени и повышения уровня продуктивности.

Среда разработки PyCharm[2] поддерживает все эти возможности, а также многие другие, такие как:

1. инструменты для запуска тестов и анализа покрытия кода, включая поддержку всех популярных фреймворков для тестирования;
2. инструменты для работы с базами данных и SQL файлами, включая удобный клиент и редактор для схемы базы данных;

3. окно классов для перехода по исходному коду по типам, а не файлам;
4. обозреватель документов для просмотра и поиска документации по продуктам на локальном компьютере или в Интернете;
5. умное автодополнение, инструменты для анализа качества кода, удобная навигация, расширенные рефакторинги и форматирование;
6. окно Свойства для настройки свойств и событий элементов управления в пользовательском интерфейсе.

Среду разработки PyCharm стоит рассматривать как интеллектуальную среду разработки, понимающую код. В процессе его написания программистом она занимается построением синтаксического дерева, определением особенностей размещенных ссылок, анализом возможных путей исполнения операторов и передачи данных.

3.3 Используемые инструменты и технологии веб-приложения

Для облегчения разработки веб-приложения нужно использовать фреймворки, которые являются набором шаблонов или заготовок, облегчающий разработку и объединение разных модулей программного проекта.

Одними из популярных фреймворков являются PHP-фреймворки Zend Framework и Symfony или Django[5], написанный на Python.

Основным критерием выбора является язык разработки серверной части, который должен корректно взаимодействовать с Фреймворком.

Анализируя различные инструменты, для разработки веб-приложения был выбран универсальный фреймворк с открытым исходным кодом Django написанный на Python3[7]. Для написания лицевой части приложения были использованы HTML[3], CSS[6], JavaScript, Bootstrap[4]. В качестве СУБД была выбрана PostgreSQL[1].

3.4 Реализация

Для работы с базой данных Django[5] использует собственную технологию программирования, в которой модель данных описывается классами Python, и по ней генерируется схема базы данных. В качестве среды разработки выбран редактор PyCharm.

В терминале PyCharm с помощью команды `python manage.py runserver` запускается сервер, выделяется адрес для localhost. В данном случае был выделен адрес 127.0.0.1:8000.

3.5 Интерфейс программы

Ниже представлен интерфейс полученного веб-приложения.

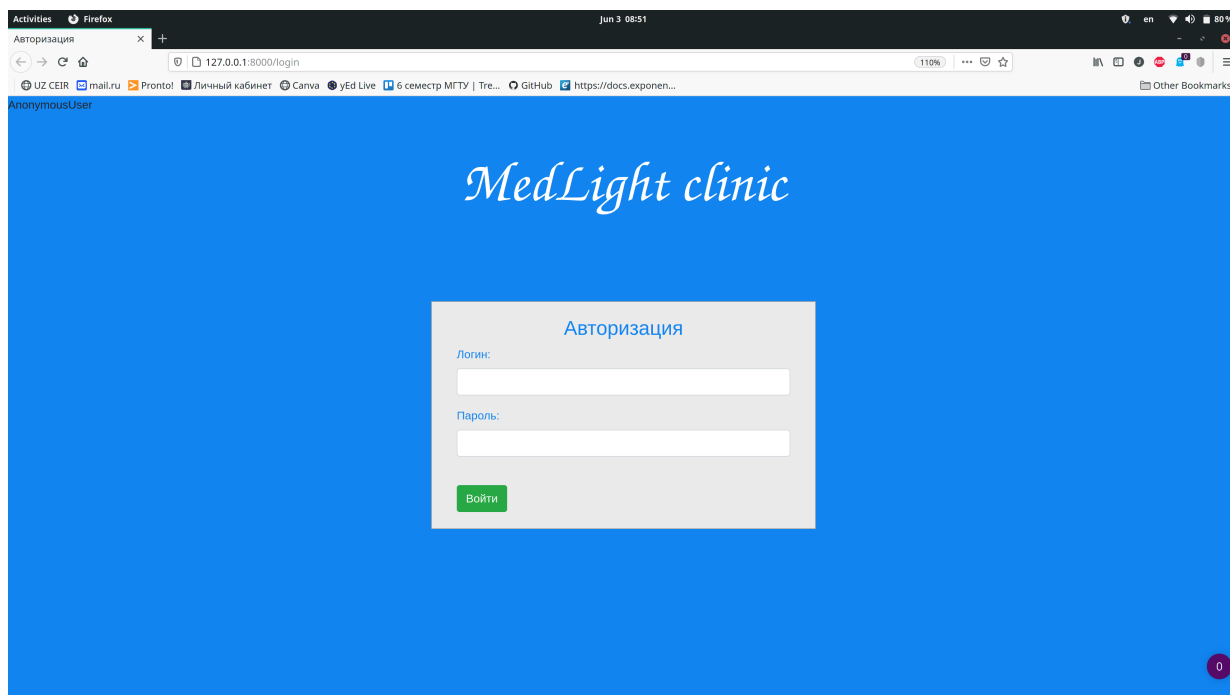


Рис. 11: Интерфейс модуля «Авторизация»

На рисунке 11 показано модуль **Авторизации**, где пользователю необходимо ввести логин и пароль от системы. В случае успеха, пользователю будет доступен функционал в соответствии с его ролью в приложении.

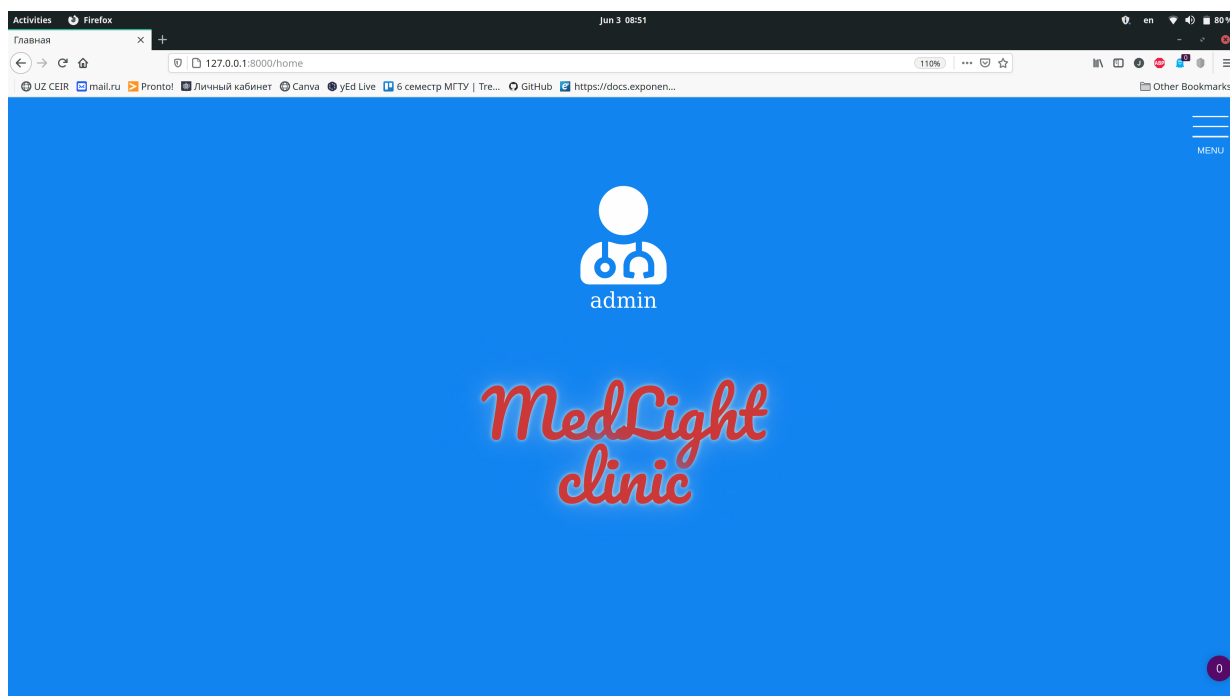


Рис. 12: Интерфейс главного меню

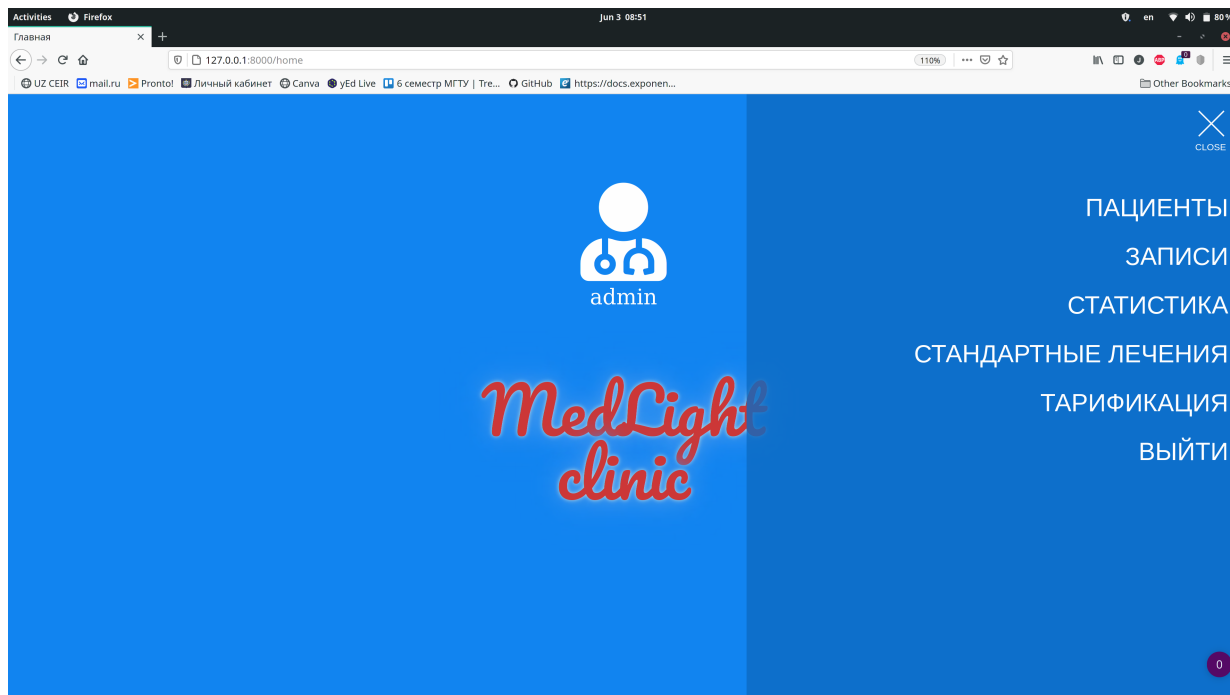


Рис. 13: Содержание меню

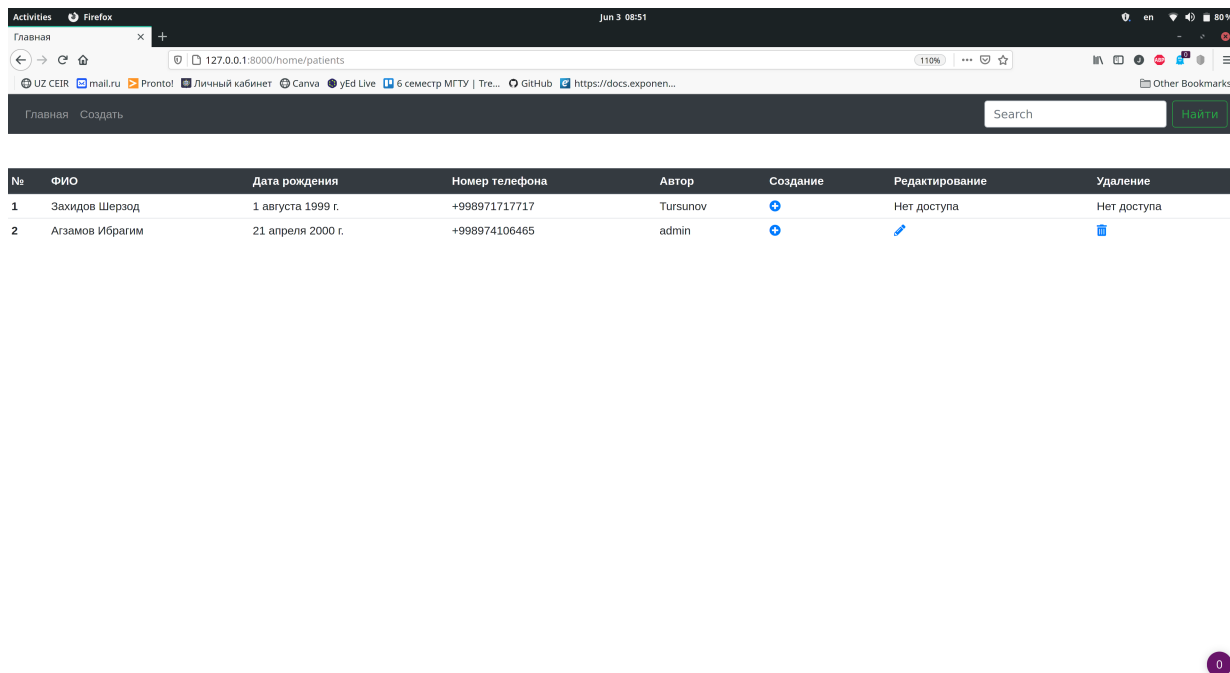


Рис. 14: Страница «Пациенты»

На этой странице, пользователю будет доступен список всех пациентов. Для каждого пациента, пользователь может создать запись у врача (Рисунок 15). Функционал удаления и изменения данных о пациенте доступен только Администратору и тем кто эти записи создал. Также на странице имеется возможность сделать поиск. Поиск работает по всем ключевым полям.

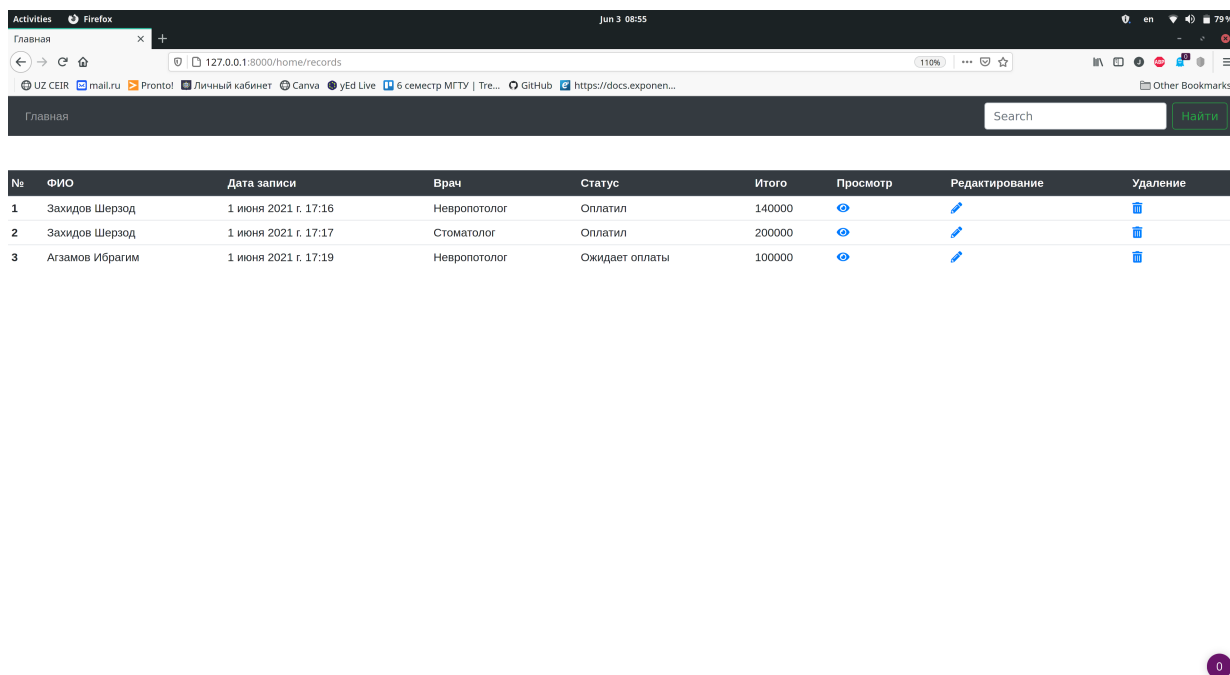


Рис. 15: Страница «Записи»

На этой странице, пользователю будет доступен список всех записей. Для каждого пациента, пользователь может просмотреть, обновить или удалить запись. Пользователю будет доступен функционал «Скачивания» рецепта на лечение, только в том случае, если была произведена оплата.

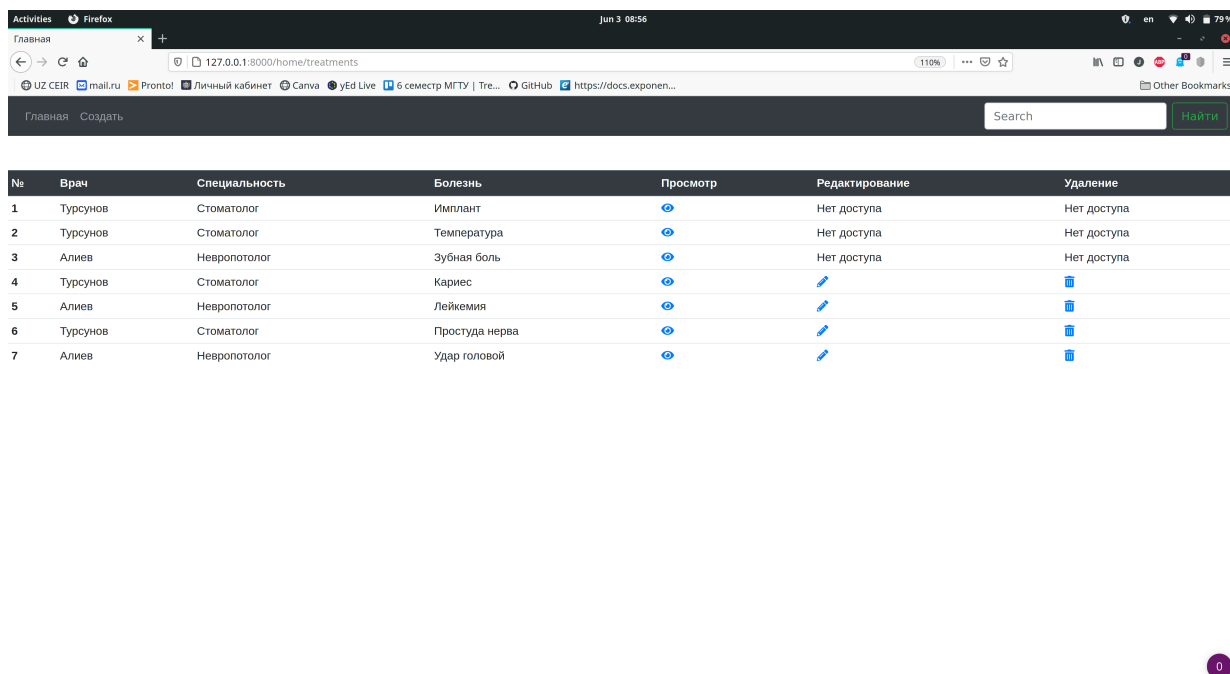
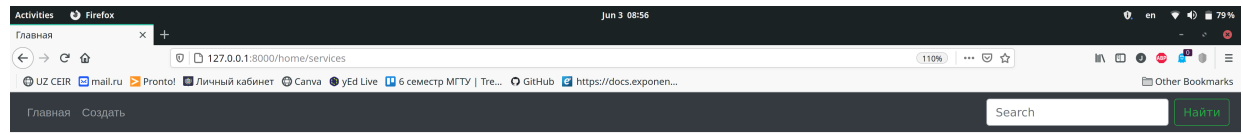


Рис. 16: Страница «Стандартные лечения»

На текущей странице пользователь может создать для себя готовое лечение, для его использования при создании записи для пациента. Этот функционал облегчает работу врачей, так как иногда приходится повторно записывать одно и то же лечение несколько раз, пациентам с одинаковым диагнозом.













№	Услуга	Цена	Редактирование	Удаление
1	ЭКГ	50000		
2	ЭЭГ	40000		
3	Осмотр Кардиолога	100000		
4	Осмотр Невропатолога	100000		
5	Удаление зуба	200000		

Рис. 17: Страница «Тарификация»

Здесь только администратор может вносить изменения для услуг, а именно изменять его название и цену.

3.6 Листинг кода

На листинге 1 представлены модели для хранения данных о пациенте и его записях у врача.

```

1  class Patients(models.Model):
2      author = models.ForeignKey(User, on_delete=models.CASCADE,
3                                 verbose_name='Author', blank=True, null=True)
4      full_name = models.CharField('FIO', max_length=50, default='')
5      date = models.DateField('Birthday', auto_now_add=False)
6      phone = models.CharField('Phone_num', max_length=13, default='None')
7
8      def __str__(self):
9          return '%s %s %s' %(self.full_name, self.date, self.phone)
10
11     class Meta:
12         unique_together = ('author', 'full_name', 'date', 'phone')
13         verbose_name = 'Patient'
14         verbose_name_plural = 'Patients'
15
16
17
18     class Records(models.Model):

```

```

19     patient = models.ForeignKey(Patients, on_delete=models.CASCADE, blank=True,
20                                null=True, related_name='records_patients')
21     author = models.ForeignKey(User, on_delete=models.CASCADE,
22                                verbose_name='Author', blank=True, null=True)
23     register_date = models.DateTimeField('Register DateTime', auto_now=True)
24     doctor = models.ForeignKey(Doctors, on_delete=models.CASCADE)
25     payment_status = models.BooleanField('Payment status')
26     used_services = models.CharField('Used Services',
27                                     max_length=70, default='')
28     total_sum = models.IntegerField('Sum')
29     disease = models.TextField('Disease', default='')
30     discharge = models.TextField('Discharge', default='')
31
32     def __str__(self):
33         return '%s %s %s %s %d' %(self.patient.full_name,
34                                   self.register_date, self.doctor.full_name,
35                                   self.payment_status, self.total_sum)
36
37     class Meta:
38         verbose_name = 'Record'
39         verbose_name_plural = 'Records'
40

```

Листинг 1: Алгоритм сортировки пузырьком

3.7 Вывод из технологической части

В данном разделе было описано и обосновано выбор языка и средств реализации динного проекта. Был продемонстрировано внутренний интерфейс программы.

4 Исследовательская часть

В данном разделе будет проведен эксперимент и сравнительный анализ. Также будут показаны примеры работы программы

4.1 Системные характеристики

Характеристики компьютера на котором проводился эксперимент:

1. операционная система - Windows 10;
2. процессор - Intel(R) Core(TM) i7-10510U CPU @1.80GHz 2.30GHz;
3. объем оперативной памяти - 16 ГБ;
4. количество ядер - 4;
5. количество логических процессов - 8;
6. видеокарта - NVIDIA GeForce GTX 1650 with Max-Q Design;
7. объем видеокарты - 4 ГБ.

4.2 Постановка эксперимента

В рамках данного проекта были проведены эксперименты, описанные ниже:

1. проверка на корректное создание, удаление, обновление данных в БД;
2. Корректная обработка в случае, если данные в БД уже существуют.

4.3 Вывод из исследовательской части

Все поставленные а данном разделе тестовые эксперименты успешно прошли проверку. Стоит отметить второй тестовый случай, когда создаваемая информация уже присутствует в БД. В данном случае срабатывает триггер, и перенаправляет пользователя на другую страницу.

Заключение

В рамках курсового проекта была реализована система контроля приема пациента в медицинском учреждении. С помощью разработанной программы можно существенно сократить время отведенное на сбор информации о пациенте и постройке логической цепочки по его истории болезни, тем самым увеличив время на его осмотр. Также приложение уменьшает нагрузку на персонал и частично помогает избавляться от бумажных документов. Были успешно реализованы следующие задачи:

1. автоматизация оформления пациента и его записи;
2. сохранение информации о пациентах – создание, редактирование и удаление;
3. минимизировать человеческий фактор при выгрузке готового рецепта на лечение;

В ходе создания приложения, со стороны медицинскогго учреждения были предложены новые возможности для данного ПО, а именно реализация очереди и возможность записать на прием врача заранее. Этот функционал будет доступен в следующей версии приложения.

Список литературы

- [1] The PostgreSQL Global Development Group. *PostgreSQL* [ЭЛ. ПЕСУРС] Режим доступа: URL: <https://www.postgresql.org/>. (дата обращения: 11.03.2021).
- [2] JetBrains. *PyCharm* [ЭЛ. ПЕСУРС] Режим доступа: URL: <https://www.jetbrains.com/ru-ru/pycharm/>. (дата обращения: 25.02.2021).
- [3] Mozilla. *htmlbook* [ЭЛ. ПЕСУРС] Режим доступа: URL: <http://htmlbook.ru/html>. (дата обращения: 13.04.2021).
- [4] Bootstrap Team. *Bootstrap* [ЭЛ. ПЕСУРС] Режим доступа: URL: <https://bootstrap-4.ru/docs/4.3.1/getting-started/introduction/>. (дата обращения: 23.04.2021).
- [5] Саймон Уиллисон Адриан Головатый. *Django Software Foundation* [ЭЛ. ПЕСУРС] Режим доступа: URL: <https://www.djangoproject.com/>. (дата обращения: 27.03.2021).
- [6] Влад Мержевич. *Основы CSS* [ЭЛ. ПЕСУРС] Режим доступа: URL: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/CSS_basics. (дата обращения: 17.04.2021).
- [7] Гвидо ван Россум. *7. Python* [ЭЛ. ПЕСУРС] Режим доступа: URL: <https://www.python.org/>. (дата обращения: 03.05.2021).