Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский университет ИТМО»

(Университет ИТМО)

Факультет Инфокоммуникационных технологий

Образовательная программа

О Т Ч Е Т

о практике Учебная, ознакомительная

Тема задания: Проектирование базы данных веб-приложения для составления графика лечения на основании рекомендаций врача.

Обучающийся: Никифоров Савелий Денисович К3220

Согласовано:
Руководитель практики от университета: Казанова Полина Петровн*а*, тьютор, Факультет инфокоммуникационных технологий

Практика пройдена с оценкой

_____

Дата _____

Санкт-Петербург
2023

СОДЕРЖАНИЕ

Стр.

ВВЕДЕНИЕ

В современном мире все больше людей обращаются к веб-сервисам для автоматизации и упрощения собственной жизни. Они органично вписались во все направления жизни людей современного общества.

Одним из таких инструментов упрощения и автоматизации жизнедеятельности может стать веб-приложение для составления графика лечения на основании рекомендаций врача. Помимо удобства для пациентов, данное приложение может помочь докторам более эффективно контроллировать процесс лечения.

Актуальность приложения обоснована следующим: по данным исследования Ассоциации директоров по коммуникациям и корпоративным медиа России (АКМР) [1] от 2023 года показывает что 78 процентов Россиян пользуются интернетом на регулярной основе, что показывает большой охват аудитории для полноценной реализации и интеграции будущего веб-приложения.

В дополнение к этому, с учетом увеличивающейся численности старшего поколения и необходимости улучшения доступа к медицинским услугам, веб-приложение подобного рода может иметь значительный потенциал для улучшения уровня здравоохранения. Это также может помочь уменьшить нагрузку на медицинский персонал, освободив их время для более критически важных задач. Кроме того, с учетом текущей ситуации в мире, веб-приложения для здравоохранения открывают новые возможности для дистанционного взаимодействия и контроля за процессом лечения, что делает их еще более актуальными и востребованными.

Целью данной практической работы является проектирование и разработка веб-приложения для составления графика лечения на основании рекомендаций врача.

Из составленной цели были выдвинуты следующие задачи:

- Анализ предметной области;
- Анализ конкурентов и аналогов приложения;
- Составление функциональных и не функциональных требований;
- Разработка базы данных приложения;
- Составление диаграм взаимодействия пользователей с системой;
- Выбор технологий программирования для создания конечного продукта;
- Создание дизайн-макет приложения;

1 Сравнение аналогов проектируемой системы

В ходе тщательного анализа и изучения предметной области, был проведен обширный поиск и исследование существующих аналогов на рынке. Данное исследование включало в себя изучение различных приложений и сервисов, их функционала, возможностей и особенностей. В результате, был получен ряд ключевых аналогов, которые в настоящее время существуют и активно используются. Эти аналоги представляют собой различные продукты и решения, которые в той или иной степени могут совпадать с поставленными целями и задачами. Изучение данных аналогов поможет лучше проанализировать существующий рынок, определить ключевые тренды и возможности для инноваций. Изученные аналоги изображены в таблице 1.1.

Таблица 1.1 — Таблица сравнения аналогов

Название Аналога	Возможности	Тариф	Достоинства	Недостатки
1	2	3	4	5
Medisafe	Мобильное приложение, которое помогает отслеживать приём лекарств по дате и времени приёма с примечанием. Также существует интеграция с приложением хранящим и обновляющим данные о здоровье и есть возможность составлять отчёт о приёме.	Бесплатное досупны базовые функции, остальные как кастомизация уведомлений и внешнего вида и отсутсвие рекламы доступны по подписке 4.99$ в месяц и 39.99$ в год.	Возможность следить за приёмом лекарств членов семьи и создание отчётности о лечении. Присутсвие примечения о способе приёма лекарства и дозировки. Интерграция с приложением следящим за показателями здоровья.	Система уведомлений не гибкая и не настраиваемая.
Mytherapy	Мобильное приложение, которое напоминает о приёме лекарств, отслеживает показатели здоровья и имеет возможность создания отчётов приёме.	Бесплатное приложение	Существование журнала о пропущенных и подтверждённых приёмов лекарств. Отслеживание количества лекарств. Поддержка схем дозировок. Выбор измерений показателей взависимости от заболевания.	Обязует следить за количеством лекарств. Не удобный выбор временит приёма.

Продолжение таблицы 1.1

1	2	3	4	5
Мои таблетки	Мобильное приложение напоминающее о приёме лекарств. Так же имеющие возможность хранить историю приёмов. И создавать примечание о приёме.	Бесплатно доступно 10 курсов приёмов лекарств. По подписке 1.99$ в месяц доступно неограниченное количество курсов приёмов.	Гибкая система уведомлений. Архив приёмов лекарств.	Отсутсвие семейного доступа и учёта пропущенных приёмов.
Mr.Pillster	Мобильное приложение напоминающие о приёме лекарств и о измерении данных о здоровье. Существует возможность создавать график приёмов и измерений. Семейный доступ к приёмам лекарств. Быстрые кнопки “принять” и “отложить” в пуш-уведомлении	Бесплатно доступно добавление только прёмов лекарств. По подписке 329 рублей раз в три месяца, 749 рублей в год и 1790 рублей единоразово доступны все функции.	Возможность создать напоминание о измерении данных о здоровье. Семейный доступ к приёму лекарств. Возможность отложить приём лекарства при появлении уведомления.	В сравнение с другими приложениями не полный список типов дозировок. Также отсутсвие возможности создать собственное примечание к приёму лекарства.
Pills time	Мобильное для напоминаний о приёме лекарств и приёмах к врачу. Семейный доступ к приёмам. Возможность добавить фото к добавленному лекарству. Гибкий выбор периодичности приёма.	Бесплатное добавление курса лекарств, без. По подписке 99 рублей в месяц в год будут доступны все функции.	Большой список примечаний связанных с едой. Семейный доступ. Возможность создать расписание приёмов у врача.	Неудобная система уведомлений. Короткий список типов дозировок.

В сравнении аналогов, были рассмотрены несколько ключевых аспектов, которые могут быть важными для пользователей, а именно:

1. Функциональность.
    - Mediasafe предоставляет широкий спектр функций, включая возможность создания индивидуального графика лечения, отслеживание прогресса и интеграцию с другими приложениями;
    - Mytherapy предоставляет следующий набор функций: составление графика лечения, гибкую настройку уведомлений, семейный доступ;
    - Мои таблетки, в отличии от других аналогов, не имеет семейного доступа, но имеет гибкую систему настройки уведомлений и архив приема лекарств;
    - Mr.Pillster предлагает более ограниченный набор функций, включая создание графика лечения и семейный доступ;
    - Pills time предоставляет только базовые функции, такие как создание графика лечения, возможность добавления заметок и семейный доступ;
2. Удобство использования.
    - Mediasafe имеет интуитивно понятный интерфейс и простую навигацию, что делает его удобным для пользователей;
    - Mytherapy имеет простой интерфейс, но некоторые функции могут быть неудобными в использовании;
    - Mr.Pillster имеет удобный интерфейс, но сложную навигацию;
    - Мои таблетки имеет интуитивно понятный интерфейс и простую навигацию, что делает его удобным для пользователей;
    - Pills time имеет сложный интерфейс, который может быть трудным для понимания и использования;
3. Стоимость.

- Mediasafe предоставляет бесплатный доступ к основным функциям, остальные доступны по подписке 4.99 $ в месяц;
- Mytherapy - бесплатное приложение;
- Мои таблетки предоставляет доступ к 10 бесплатным курсам лечения, а для неограниченного доступа необходима подписка 1.99 $ в месяц;
- Mr.Pillster предоставляет бесплатный доступ к добавлению приемов лекарств, для разблокирования других функций необходима подписка, которая стоит 3.6 $ раз в 3 месяца;
- Pills time предоставляет бесплатный доступ к добавлению приемов лекарста, для разблокирования других функций необходима подписка 1 $ в месяц;

Исходя из сравнения, можно сделать вывод, что аналог Mytherapy является наиболее предпочтительным выбором для пользователей, которые ищут веб-приложение для составления графика лечения на основании рекомендаций врача. Он предлагает широкий спектр функций, удобный интерфейс и польностью бесплатное использование, поэтому, при дальнейшем анализе, стоит опираться на этот аналог.

2 Анализ функицональных и нефукциональных требований

В данной главе будет представлен анализ будущего приложения с точки зрения выделения функциональных и нефукциональных требований.

2.1 Функциональные требования

В процессе анализа требований к веб-приложению для составления графика лечения на основе рекомендаций врача были выделены следующие фукнциональные требовация, которые необходимы для полноценного функционирования приложения:

1. Веб-приложение должно позволять пациенту или его лечащему врачу создавать график лечения на основании рекомендаций и назначений. Для этого необходимо предоставить пользователю возможность добавлять информацию о лекарствах, дозах и промежутках времени приема, а также продолжительность лечения. График должен быть составлен на необходимый период времени и сохраняться для каждого пациента.
2. Приложение должно иметь функцию добавления рекомендаций и назначений, включая лекарства, дозы и промежутки времени приема.
3. Приложение должно позволять врачу указывать продолжительность лечения, чтобы график был составлен на необходимый период времени.
4. Приложение должно предоставлять возможность сохранять и редактировать график лечения для каждого пациента, чтобы врач мог обновлять его при необходимости.
5. Приложение должно иметь функцию проверки наличия противопоказаний и взаимодействия между различными лекарствами, чтобы врач мог быть уверен в безопасности назначенного лечения.

6. Приложение должно предоставлять возможность врачу проверить, были ли выполнены назначения пациентом, путем отметки о выполнении каждой рекомендации.
7. Приложение должно создавать напоминания для пациента о приеме лекарств и выполнении других назначений, которые были добавлены самим пациентом или его лечащим врачом.
8. Приложение должно обеспечивать безопасное хранение конфиденциальной информации о пациентах и доступ только для авторизованных врачей.

2.2 Нефункциональные требования

После формулирования функциональных требований к системе и анализа системы с точки зрения взаимодействия с конечным потребителем, были выделены следующие нефункциональные требования:

1. Веб-приложение должно иметь интуитивно понятный и привлекательный интерфейс для пользователей всех возрастов и уровней навыков.
2. Приложение должно поддерживать различные устройства и разрешения экранов, включая мобильные устройства, планшеты и компьютеры.
3. Гарантия конфиденциальности и безопасности персональных медицинских данных пациентов. Так как данный аспект строго регулируется государством.
4. Обеспечить совместимость с различными современными бразуерми, при помощи который будет исполняться взаимодействие с приложением для удобства пользователей.
5. Обеспечить быструю загрузку и отзывчивость интерфейса при работе с графиком лечения и рекомендациями врача.

6. Поддерживать возможность локализации интерфейса на русском языке для удовлетворения потребностей пользователей на территории Российской Федерации.
7. Реализовать авторизацию и управление доступом к приложению и данным для обеспечения безопасности и ограничения доступа к конфиденциальным данным.
8. Обеспечить отказоустойчивость для минимизации времени простоя и потери данных в случае сбоев или неполадок. Приложение должно быть обеспечено необходимыми механизмами для минимизации времени простоя и потери данных в случае возникновения сбоев или неполадок. Это означает, что система должна быть способна восстанавливаться после сбоев без значительного воздействия на функциональность и производительность, обеспечивая непрерывность предоставления услуг пользователям.

3 Диаграммы прецедентов

В данном разделе будут представлены взаимодействия основных пользователей и ролей в системе в виде диаграмм прецедентов и схем активности. Для создания изображений диаграмм было использовано бесплатное программное обеспечения под названием "diagrams.net". [2]

3.1 Диаграмма прецедентов для сущности "Пользователь"

Сущность "Пациент" имеет следующие возможности:

1. Добавить прием лекарства: Эта возможность позволяет пациенту добавить информацию о приеме лекарства. В процессе добавления создается уведомление, которое отправляется пациенту телеграмм-ботом в определенное время, заданное пациентом;
2. Добавить визит врача: Эта возможность позволяет пациенту добавить информацию о визите к врачу. В процессе добавления создается уведомление, которое отправляется пациенту телеграмм-ботом в определенное время, заданное пациентом;
3. Регистрация пациента: Эта возможность позволяет пациенту зарегистрироваться в системе. В процессе регистрации создается экземпляр пациента в базе данных;
4. Редактирование профиля: Эта возможность позволяет пациенту редактировать свои данные в профиле. В процессе редактирования обновляются данные в базе данных;
5. Отправка обратной связи: Эта возможность позволяет пациенту отправить обратную связь о системе или услугах. В процессе отправки обновляются данные в базе данных;

Таким образом, сущность "Пациент" имеет пять возможностей, некоторые из которых включают в себя создание уведомления и их отправку телеграмм-ботом, а также обновление данных в базе данных. Диаграмма сущностей изображена на рисунке 3.1.

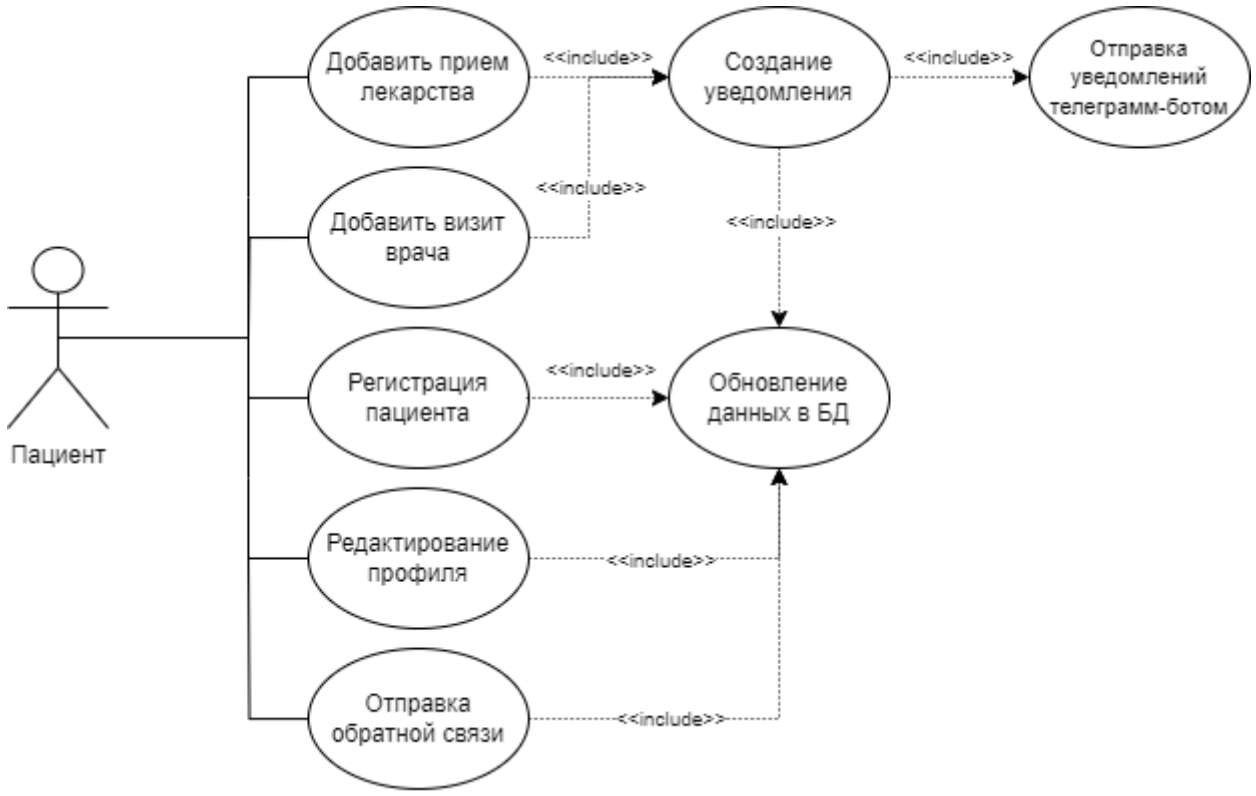

Рисунок 3.1 — Диаграмма прецедентов для роли "Пациент".

3.2 Диаграмма прецедентов для сущности "Доктор"

Сущность "Доктор" имеет следующие возможности:

1. Добавить график лечения пациентов: Эта возможность позволяет доктору добавлять информацию о приемах лекарств для пациента. В процессе добавления создается уведомление, которое отправляется пациенту телеграмм-ботом в определенное время, заданное доктором;
2. Назначить прием: Эта возможность позволяет доктору назначить прием с пациентом. В процессе добавления создается уведомление, которое

отправляется пациенту телеграмм-ботом в определенное время, заданное доктором;

3. Регистрация доктора: Эта возможность позволяет доктору отправить заявку на регистрацию модератору веб-приложения. В процессе регистрации создается экземпляр доктора с атрибутом, показывающий, что доктор не прошел авторизацию и не подал заявку на подтверждение данных для модератора в базе данных;
4. Редактирование профиля: Эта возможность позволяет доктору редактировать свои данные в профиле. В процессе редактирования обновляются данные в базе данных;
5. Получить список пациентов: Эта возможность позволяет доктору получить список лечащихся и уже прошедших лечение пациентов;

Итоговая диаграмма прецедентов изображена на рисунке 3.2.

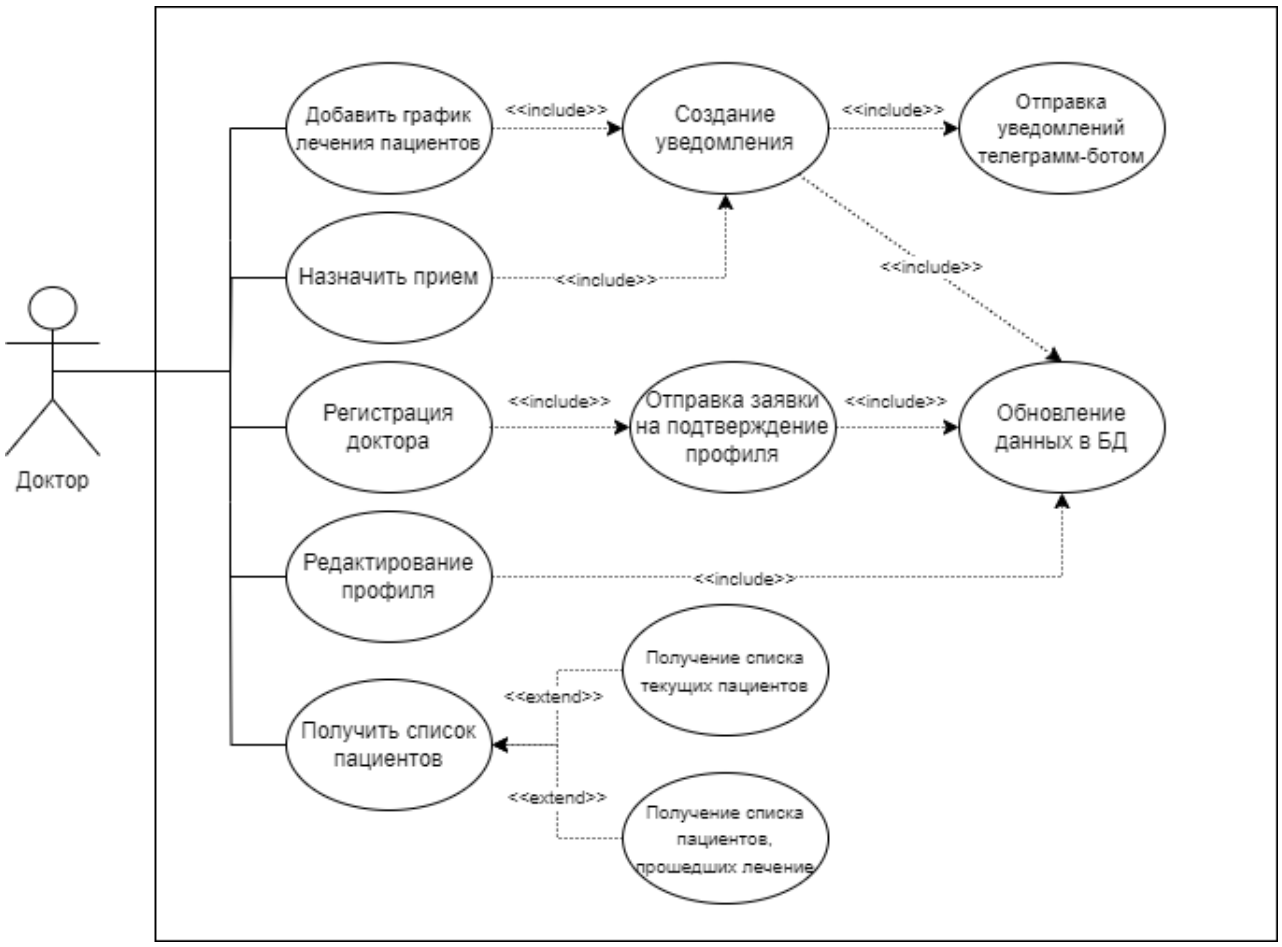

Рисунок 3.2 — Диаграмма прецедентов для роли "Доктор".

3.3 Диаграмма прецедентов для сущности "Менеджер"

Сущность "Менеджер” имеет следующие возможности:

1. Получение отзывов о врачах: Эта возможность позволяет менеджеру просмотреть статистику доктора, такую как количество принятых пациентов, проведенных консультаций, выписанных рецептов и других показателей, которые могут помочь оценить эффективность работы доктора. Менеджер может использовать эту информацию для анализа производительности доктора, определения его сильных и слабых сторон, а также для принятия решений о его дальнейшем развитии и обучении. Эта функция также может быть полезна для определения потребностей

пациентов и улучшения качества медицинских услуг, предоставляемых клиникой;

2. Администрирование данных о врачах: Эта возможность позволяет менеджеру подтверждать, отклонять заявку о регистрации доктора, а также, при необходимости, отстранять доктора от врачебной деятельности. Все это влечет за собой обновление данных в БД;
3. Изменение данных пользователя. Это возможность позволяет менеджеру редактировать данные пациента, которые были указаны при регистрации. При изменении данных пользователя, система автоматически сохраняет новые значения в базе данных, чтобы они были доступны для системных процессов, которые могут нуждаться в этой информации;

Итоговая диаграмма прецедентов изображена на рисунке 3.3.

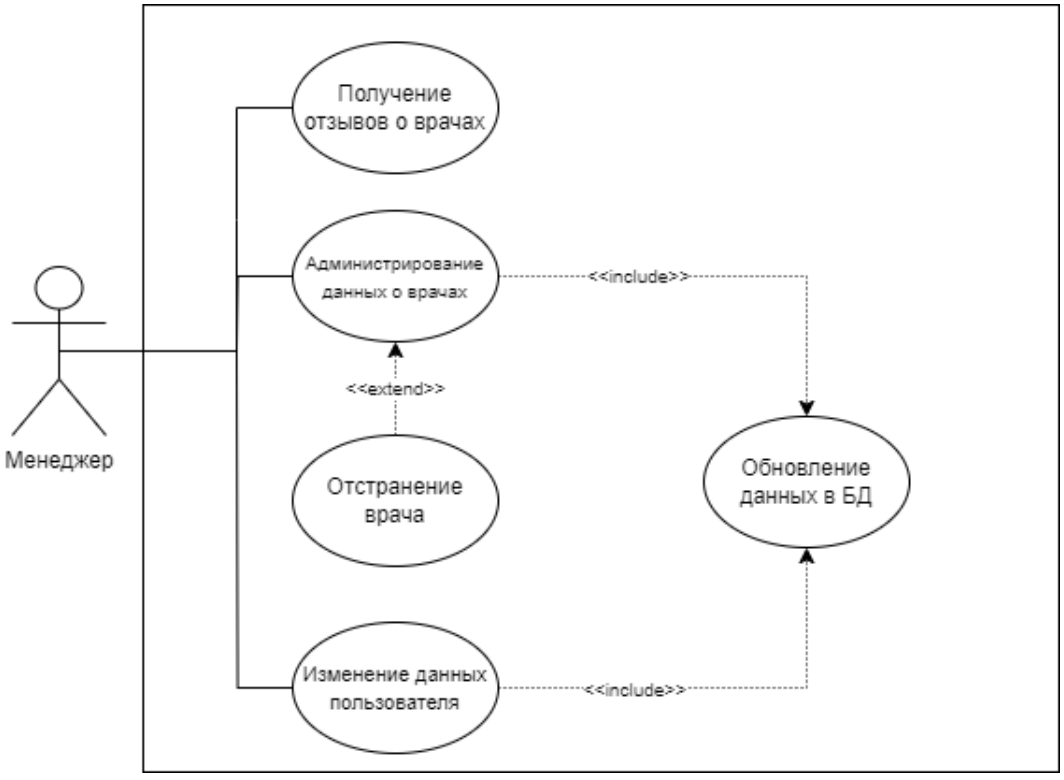

Рисунок 3.3 — Диаграмма прецедентов для роли "Менеджер".

4 Проектирование базы данных приложения

В данной главе будет представлен процесс проектирования базы данных веб-приложения для составления графика лечения на основании рекомендаций врача.

4.1 Потенциальные объекты системы

В процессе анализа предметной области были выделены следующие примерные бизнес-сущности, которые должны быть отображены в базе данных:

- Доктор;
- Пациент;
- Менеджер;
- Прием лекарств;
- Прием врача;
- Лекарственное средство;
- Рекомендации;
- Отзыв;

4.2 Определение атрибутов и первичных ключей

Для оптимизации работы веб-приложения, было принято решение о денормализации базы данных, объединив объекты системы в сущность "Пользователь". Для реализации контроля доступа внутри системы будет введен атрибут роли, который будет ограничивать доступ к изменению и получения данных из системы.

В таблице 4.1 представлены атрибуты сущности "Пользователь".

Таблица 4.1 — Описание атрибутов сущности "Пользователь".

Название атрибута	Обязательный/не обязательный (*/o)	Уникальный идентификатор (#)	Тип для логической модели
idUser	*	#	Числовой
email	*	(#)	Символьный
role	*		Символьный
firstName	*		Символьный
secondName	*		Символьный
lastName			Символьный
phoneNumber			Символьный
idTelegram			Числовой
idLicense			Символьный
isConfirmed			Логический

Поле "idTelegram" необходимо для авторации и рассылки уведомлений через телеграмм-бота, который будет напоминать врачам и пользователям о изменении их лечебного плана и о приеме необходимых лекарственных препаратов.

Поля "idLicense" и "isConfirmed" относятся к роли врача. "idLicense" относиться к номеру идентифицирующему документ об образовании врача. Поле "isConfirmed" показывает был ли допущен доктор к работе с пациентами после проверки документов менеджером.

Взаимодействие между пациентом и доктором реализуется при помощи сущности пересечения, которая показывает какие врачи производят наблюдения и рекомендации по лечебному плану для определенного пользователя.

В таблице 4.2 представлены атрибуты сущности пересечения между ролями системы "Пациент" и "Доктор".

Таблица 4.2 — Описание атрибутов сущности пересечения ролей "Пациент" и "Доктор".

Название атрибута	Обязательный/не обязательный (*/o)	Уникальный идентификатор (#)	Тип для логической модели
idUser	*	#	Числовой
idDoctor	*	#	Числовой

Для реализации уведомлений, была выделена сущность "Прием лекарств". Данная сущность дожна хранить в себе атрибуты, для реализации следующих видов приема:

- Прием только один день;
- Ежедневный прием;
- Прием каждые X дней;
- Прием по конкретным датам;
- Цикл приема из X дней и Y дней пропуска;
- Прием через определенный промежуток времени;

Также необходимо хранить время приема в течении дня для повышения эффективности лечения.

В таблице 4.3 представлены атрибуты сущности "Прием лекарств".

Таблица 4.3 — Описание атрибутов сущности "Прием лекарств".

Название атрибута	Обязательный/не обязательный (*/o)	Уникальный идентификатор (#)	Тип для логической модели
idPill	*	#	Числовой
idUser	*		Числовой
type	*		Символьный
name	*		Символьный
form			Символьный
note			Симовльный
startDate	*		Даты и времени
endDate			Даты и времени
time			Временной
days			Дата
deltaTakeDays			Числовой
deltaPassDays			Числовой
timeDelta			Временной
lastNotify			Даты и времени

Для отслеживание удовлетворенности пользвателей в услугах выбранного лечащего врача была введена сущность "Отзыв". Данная сущность предназначена для администрирования площадки менеджером и отслеживании качества предоставляемых докторами услуг. Атрибуты сущности "Отзыв" представлены в таблице 4.4

Таблица 4.4 — Описание атрибутов сущности "Отзыв".

Название атрибута	Обязательный/не обязательный (*/o)	Уникальный идентификатор (#)	Тип для логической модели
idComment	*	#	Числовой
idPatient	*		Числовой
idDoctor	*		Числовой
rate	*		Числовой
review			Символьный

Для эффективного процесса лечения также необходимы своевременное посещение лечащего врача. Для этого в системе необходимо хранить информацию о плановых посещениях, которые представлены в сущности "Прем врача". В таблице 4.5 представлены атрибуты сущности "Прием врача".

Таблица 4.5 — Описание атрибутов сущности "Прием врача".

Название атрибута	Обязательный/не обязательный (*/o)	Уникальный идентификатор (#)	Тип для логической модели
idVisit	*	#	Числовой
idPatient	*		Числовой
idDoctor			Числовой
appointmentTime	*		Даты и времени
note			Символьный

Для предоставления комфортного пользования сервисом были выделены справочные сущности, которые позволяют пользователю ознакомиться с препаратами и предоставить поддержку советами о правильном лечении.

Сущность "Лекарственное средство" является справочной сущностью возможных лекартсвенных препаратов. Аттрибуты данной сущности представлены в таблице 4.6

Таблица 4.6 — Описание атрибутов сущности "Лекарственное средство".

Название атрибута	Обязательный/не обязательный (*/o)	Уникальный идентификатор (#)	Тип для логической модели
idPillType	*	#	Числовой
name	*		Символьный
dose	*		Числовой
form	*		Символьный
description			Символьный

Сущность "Совет" является справочной сущностью, которая должна помочь пользователю придерживаться эффективного плана лечения. Атрибуты сущности "Совет" представлены в таблице 4.7.

Таблица 4.7 — Описание атрибутов сущности "Совет".

Название атрибута	Обязательный/не обязательный (*/o)	Уникальный идентификатор (#)	Тип для логической модели
idTips	*	#	Числовой
title	*		Символьный
content	*		Символьный

4.3 Логическое проектирование базы данных

Логическое проектирование базы данных веб-прилоежения было произведено при помощи программы "PgAdmin" [3]. Данный комплекс программного обеспечения является обширным инструментом для логического проектирования и администрирования баз данных. Оно предоставляет возможность создания ERD (Entity-Relationship Diagram) диаграмм, которые представляют структуру базы данных и связи между ее элементами. "PgAdmin" также позволяет конвертировать ERD диаграммы в SQL-скрипты, которые можно использовать для создания таблиц в базе данных. Это значительно упрощает процесс разработки. Результат преобразования сущностей в ERD диаграму изображен на рисунке 4.1.

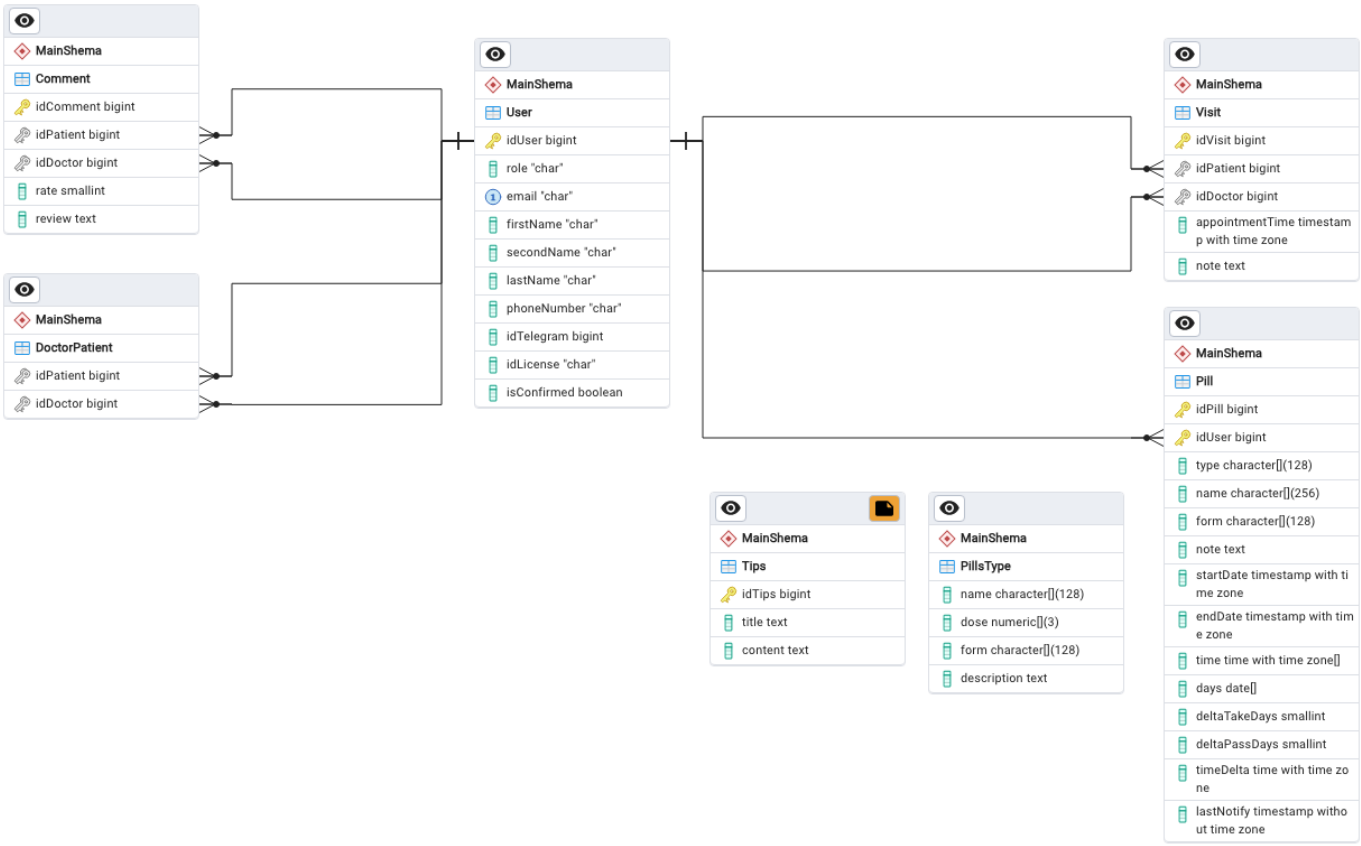

Рисунок 4.1 — ERD диаграмма базы данных.

5 Алгоритмы работы с пользователем

В данном разделе будут представлены алгоритмы работы с потенциальным пользователем веб-приложения.

5.1 Диаграммы активности

Диаграмма активности “Регистрация пациента” представляет собой последовательность следующих действий:

1. Пользователь отправляет данные для регистрации;
2. Происходит проверка введенных данных, в случае если были предоставлены некоректные данные, пользователь вводит данные снова, а в случае успешной проверки - происходит следующая этап взаимодейсвия;
3. Происходит проверка на уникальность пациента, в случае не уникальности пациента - пользователь вводит данные снова, в ином случае - происходит регистрация пользователя;

Диаграмма активности “Регистрация пациента” изображена на рисунке 5.1.

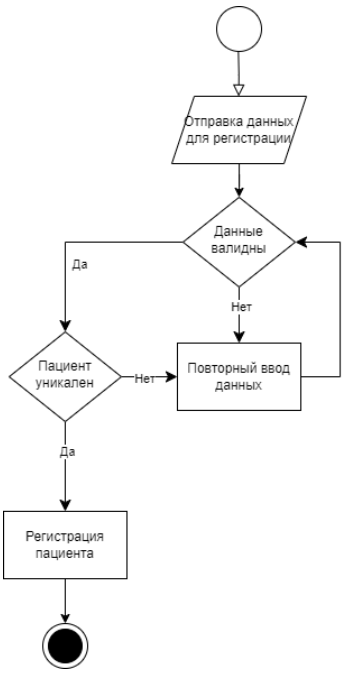

Рисунок 5.1 — Диаграмма активности “Регистрация пациента”.

Диаграмма активности “Получить список пациентов” представляет собой последовательность следующих действий:

1. Пользователь отправляет запрос на получение списка пациентов;
2. Происходит проверка на существование данных, в случае существовании данных происходит следующий этап - получение списка пациентов, в ином случае - происходит извещение доктора об отсутствии пациентов;

Диаграмма активности “Получить список пациентов” изображена на рисунке 5.2.

Рисунок 5.2 — Диаграмма активности “Получить список пациентов”.

Диаграмма активности “Администрирование данных о врачах” представляет собой последовательность следующих действий:

1. Менеджер отправляет запрос на получение данных о врачах;
2. Происходит проверка корректности данных, если данные корректны, то происходит следующий этап, в ином случае происходит отказ в доступе к пациентам для врача;
3. Происходит проверка отзывов лечащего врача, оставленных пациентами, в случае, если отзывы приемлемы врач утверждается, в ином случае - происходит отказ в доступе к пациентам для врача;

Диаграмма активности “Администрирование данных о врачах” изображена на рисунке 5.3.

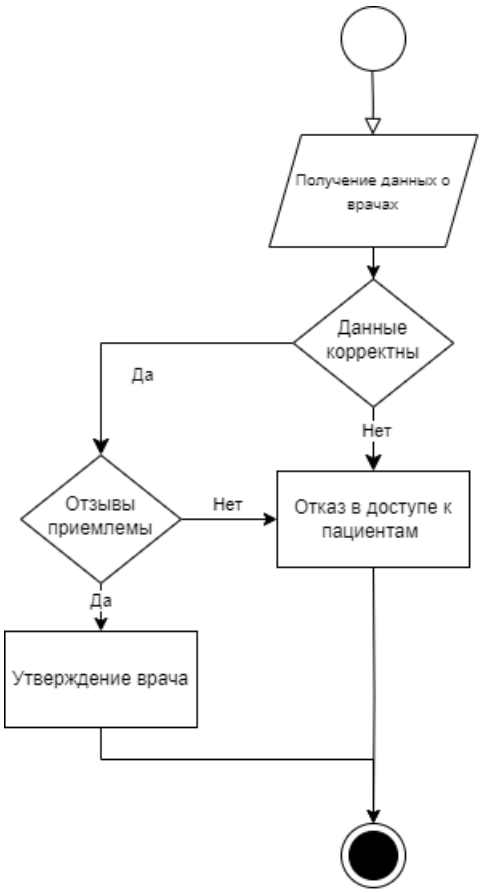

Рисунок 5.3 — Диаграмма активности “Администрирование данных о врачах”.

5.2 Контекстные диаграммы взаимодейсвия

В данном подразделе представлена модель в нотации IDEF3, которая описывает информационные потоки и взаимоотношения между процессами обработки информации и объектами, являющимися частью этих процессов. Модель представлена в виде контекстной диаграммы (рисунок 5.4), которая представляет собой графическое представление основных компонентов системы и их взаимосвязей и позволяет определить основные процессы и объекты, которые участвуют в обработке информации, а также информационные потоки между ними.

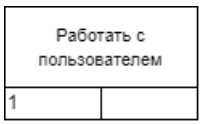

Рисунок 5.4 — Контекстная диаграмма (IDEF3)

Декомпозиции этой диаграммы изобажена на рисунке 5.5. Декомпозиция представляет собой более детальное представление процессов и объектов, описанных в контекстной диаграмме, позволяет более подробно описать каждый процесс и объект, а также их взаимосвязи. Кроме того, она включает в себя описание входных и выходных данных, используемых ресурсов и исполнителей, необходимых для выполнения процессов.

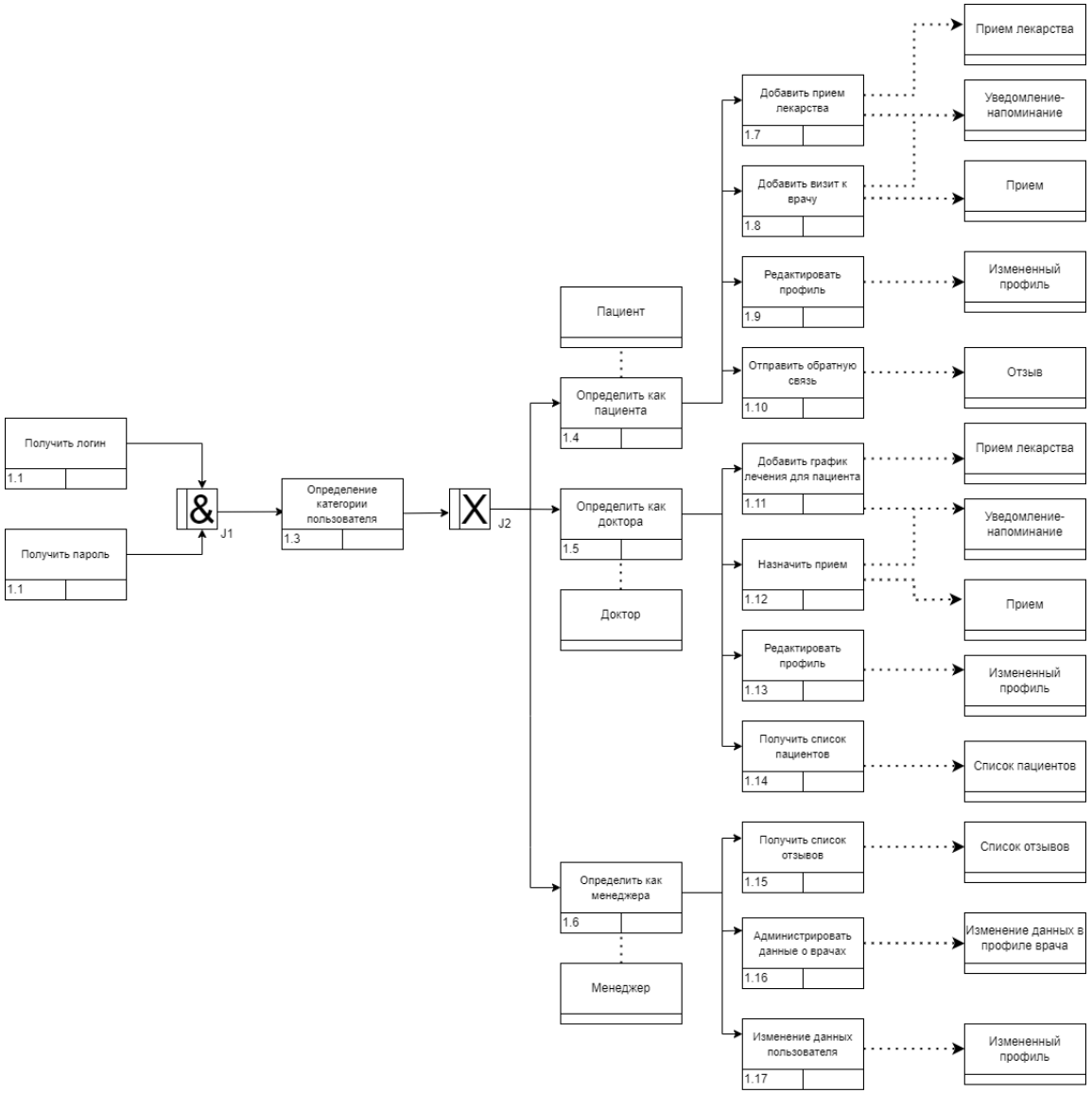

Рисунок 5.5 — Декомпозиция контекстной диаграммы

Декомпозиции блока “Администрировать данные о врачах” представлена на рисунке 5.6

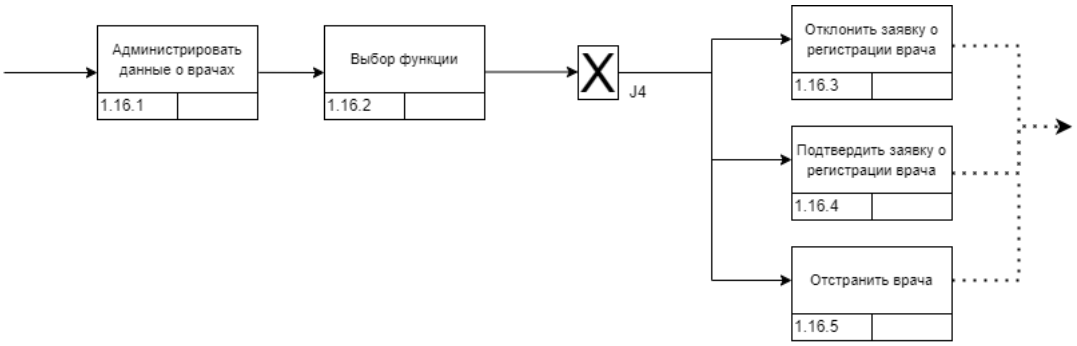

Рисунок 5.6 — Декомпозиции блока “Администрировать данные о врачах”

Декомпозиция блока Декомпозиция блока “Получить список пациентов” представлена на рисунке 5.7

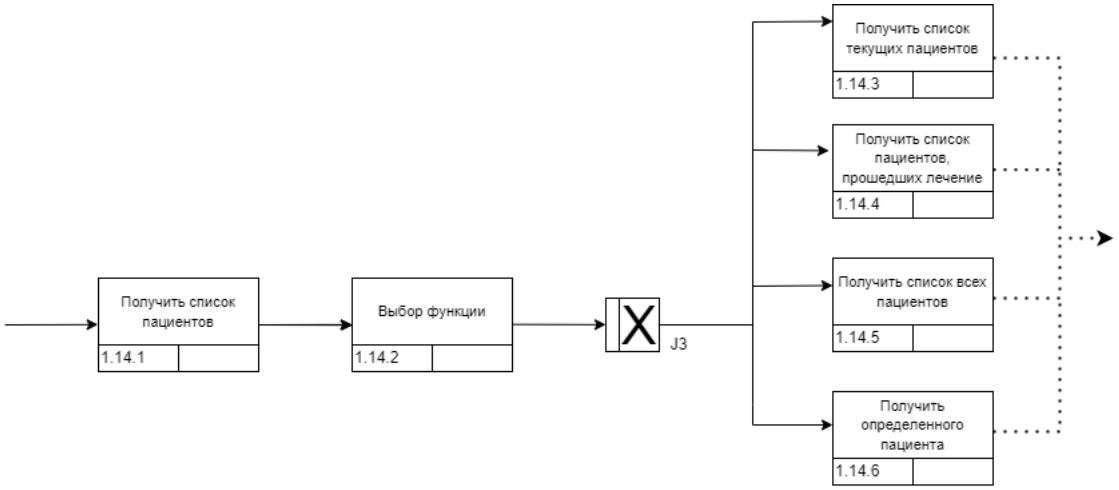

Рисунок 5.7 — Декомпозиции блока “Администрировать данные о врачах”

Декомпозиция блока “Редактировать профиль” представлена на рисунке 5.8

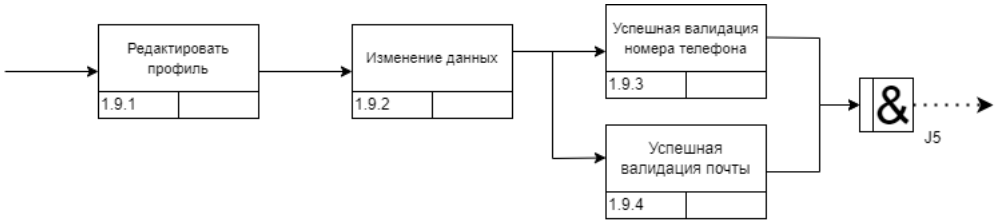

Рисунок 5.8 — Декомпозиции блока “Администрировать данные о врачах”

6 Технологии программирования

В данной главе будут представлены выбранные технологии программироания, которые будут применены при разработке веб-приложения для составления графика лечения на основании рекомендаций врача.

6.1 Веб-приложение

Для реализации клиент-серверной архитектуры будут применяться следующие технологии программирования:

1. Python - [4] это интерпретируемый язык программирования, который широко используется в веб-разработке. Он имеет простой и понятный синтаксис, множество библиотек для создания веб-приложений и хорошо масштабируется.
2. Django [5] - это обширная библеотека для языка Python, которая предоставляет множество инструментов для разработки веб-приложений. Она обеспечивает высокую производительность, безопасность и масштабируемость. Django также имеет встроенную поддержку для работы с базами данных, шаблонизации и аутентификации.
3. JavaScript [6] - это интерпретируемый язык программирования, который широко используется для разработки веб-приложений. Также нативно поддерживается современными браузерами.
4. React [7] - это библиотека для создания пользовательских интерфейсов на языке JavaScript. Она предоставляет мощные инструменты для создания интерактивных и высокопроизводительных веб-приложений. React позволяет легко управлять состоянием приложения и обновлять его части, что делает его идеальным выбором для разработки сложных интерфейсов.

6.2 Система уведомлений пользователей

Для реализации системы уведомлений пользователей было принято решение воспользоваться телеграмм-ботом для рассылки уведомлений. Данный выбор связан с обширной базой пользователей телеграма [8].

Для программной реализации данной концепции была выбрана библиотека Aiotelegram [9] Данная библиотека предназначена для работы с интерфейсом программирования приложения (API) Telegram. Она позволяет разработчикам создавать ботов для Telegram и применять интерактивные пути взаимодействия с пользователем. В данном случае будет создан телеграмм-бот для отправки уведомлений пациентам.

6.3 Развертывание приложения

Для успешной эксплуатации приложения, планируется использовать современные технологии развертывания приложений. Для достижения гибкости использования предоставленной провайдером серверной инфраструктуры будет использоваться инструмент контейнеризации приложений под названием Docker [10]. Так как приложения является небольшим, то для оркестрации данных контейнеров будет использоваться поставляемый вместе с Docker инструмент Docker-compose [11]. Использование данного подхода технологий позволит создать надежное и масштабируемое веб-приложение для составления графика лечения на основании рекомендаций врача.

7 Диаграммы классов

После определения технологий программирования был начат этап разработки диаграммы классов. В процессе исследования хороших практик разработки веб-приложений было принято решение использовать паттерн проектирования “Модель-Представление-Контроллер”. Данный паттерн был выбран, потому что он позволяет логически разделить функции приложения на компоненты, что упрощает работу с проектом. Кроме того, данный паттерн позволяет создать динамичный веб-сайт, это является преимуществом, так как данный проект должен адаптироваться под пользователя.

В выбранной библиотеки для разработки приложения данный паттерн проектирования реализуется при помощи следующих базовых классов:

- Моделью в библиотеке “Django” представляется класс “Model”. Как правило, модель наследуется от данного базового класса и содержит атрибуты, представляющие поля базы данных, а также методы для работы и обработки информации, которые содержаться в атрибутах класса.
- Контроллером в библиотеке представляется класс “Serializer”, в частности класс “ModelSerialzier”. Данный класс позволяет интерпретировать действия пользователя оповещая модель о необходимости изменения.
- Представлением является класс “APIVIew”. Данный класс позволяет обрабатывать HTTP запросы и возвращать ответ в формате JSON при помощи сериализаторов, которые выступают в роли контроллера.

7.1 Составление диаграммы классов моделей

Использование библиотеки “Django” [5] для разработки веб-приложения позволяет использовать предоставленный базовый класс модели, который

реализует механизм объектно-реляционного отображение (ORM) классов. В соответсвии с разработанной схемой базы данных были спроектированы классы-модели. Диаграмма классов предоставлена на рисунке 7.1.

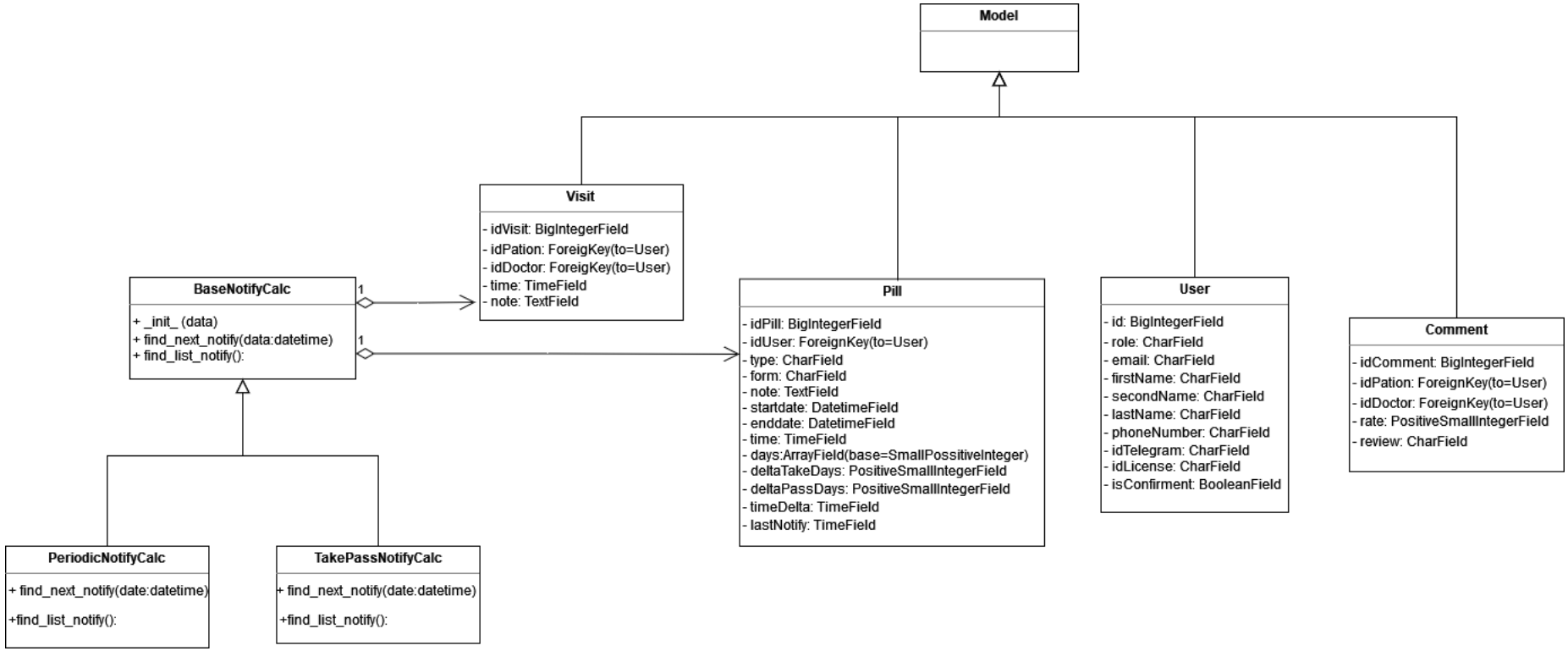

Рисунок 7.1 — Диаграмма классов моделей.

Каждый представленный на диаграмме 7.1 отображает таблицу в базе данных. От класса “Model” наследуются “Pill”, “Visit”, “User”, “Comment”. Данное наследование обоснованно необходимостью использования механизма объектно-реляционного отображения. Также, каждый атрибут перечисленных классов является классом, который предоставляется в библиотеке “Django” [5], для корректного преобразования информации из базы данных.

Для реализации логики нахождения времени уведомления в класс “Pill”, который отображает сущность “Прием препаратов”, был добавлен атрибут “data_calc”. Данный атрибут является классом-фабрикой. Данный класс возвращает необходимый класс калькулятора для нахождения времени уведомления для пользователя на основании типа приема. Класс “BaseNotifyCalc” содержит в себе методы “find_next_notify” и “find_list_notify”. Эти методы возвращают следующую дату уведомления и список дат, когда должно прийти уведомление о приёме лекарства. В следствии эти методы будут перегружены в классах наследниках “PeriodNotifyCalc” и “TakePassNotifyCalc”. В родительском классе происходит расчет дат уведомлений, исходя из того, что пациент принимает лекарство каждый день в определённое время. В классах “PeriodNotifyCalc” и “TakePassNotifyCalc” - пациент принимает лекарство через фиксированное количество дней в определённое время, или пациент принимает лекарство подряд несколько дней и делает паузу на определённое количество дней соответственно.

7.2 Составление диаграммы классов контроллеров

Благодаря библиотеке “Django” были спроектированы классы выполняющие функцию сериализации при помощи наследования от класса “ModelSerialiser”, который предоставляет возможность создать контроллеры, работающие с экземплярами моделей и наборами запросов.

В соответствии с разработанными классами-моделями были созданы классы для сериализации. Диаграмма классов изображена на рисунке 7.2.

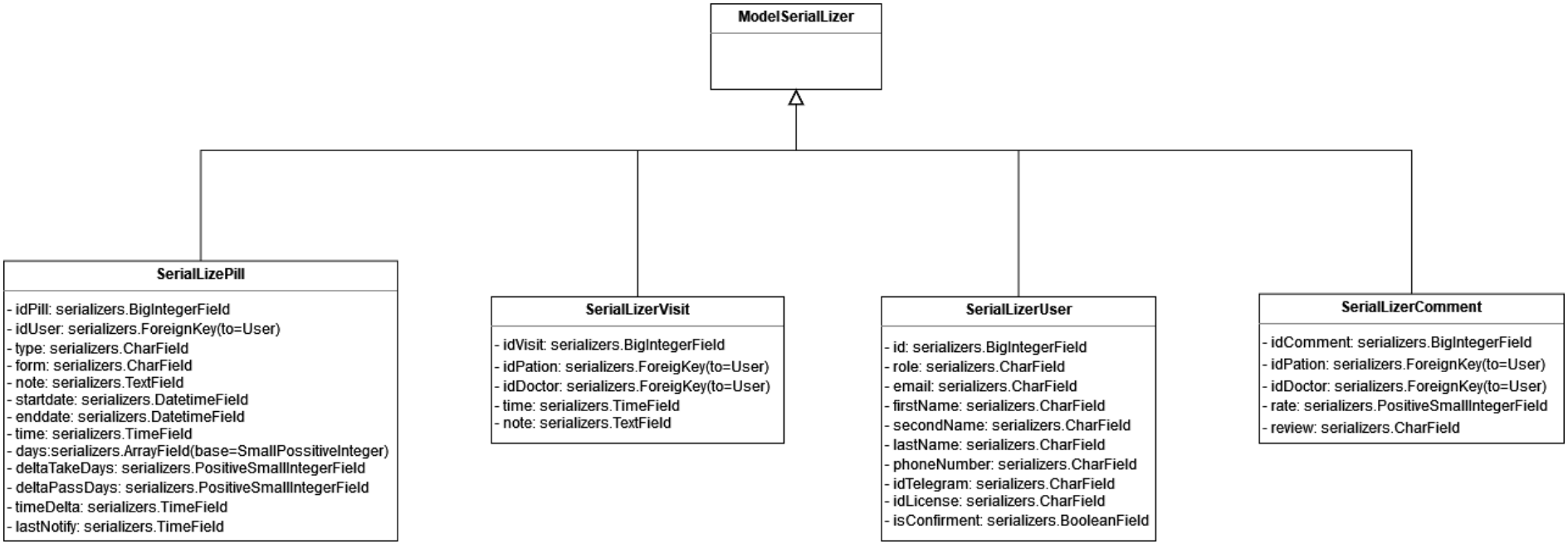

Рисунок 7.2 — Диаграмма классов контроллеров.

Классы сериализаторов “SerialLizePill”, “SerialLizerUser”, “SerialLizerVisit”, “SerialLizerComment” созданы для сериализации данных классов “Pill”, “User”, “Visit”, “Comment” в JSON или XML. Так же стоит заметить, что для всех атрибутов используется Field класса “Serializer” для представления отношений, в которых один тип объекта вложен в другой. Для указанных вложенных представлений будут указаны флаги “required=False” и “many=True” в случае, если вложенное представление может опционально принимать значение None или если вложенное представление должно быть списком элементов соответственно.

7.3 Составление диаграммы классов представлений

Функцию представлений в используемой библиотеке выполняет класс ViewSet, который позволяет обрабатывать все запросы, связанные с операциями удаления, получения, изменения и обновления модели при помощи контроллера. Данный класс реализуется при помощи наследования от базового класса “ViewsSet” и указанием используемого контроллера для взаимодействия и интерпретирования пользовательских действий, который изменяют состояние используемой модели. Диаграмма классов изображена на рисунке 7.3.

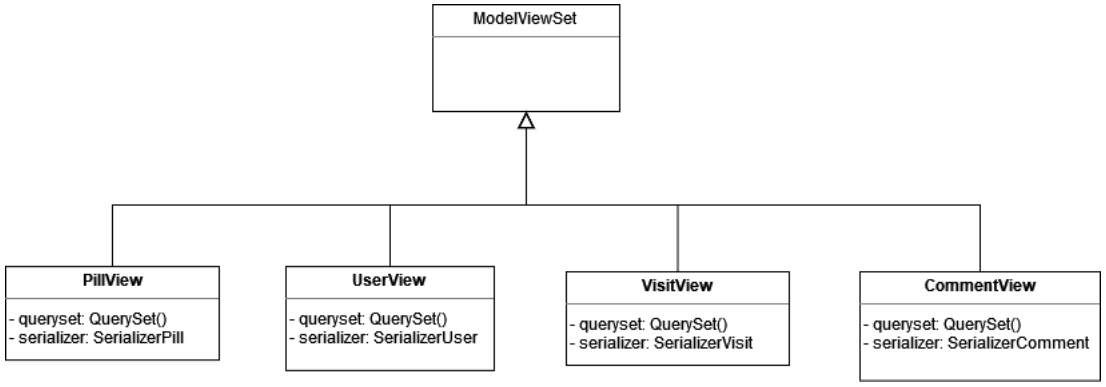

Рисунок 7.3 — Диаграмма классов представлений.

7.4 Составление диаграммы классов сериализации и представлений

В указанных выше пунктах были созданы классы представлений и сериализации. В результате была создана итоговая диаграмма. Диаграмма классов изображена на рисунке 7.4.

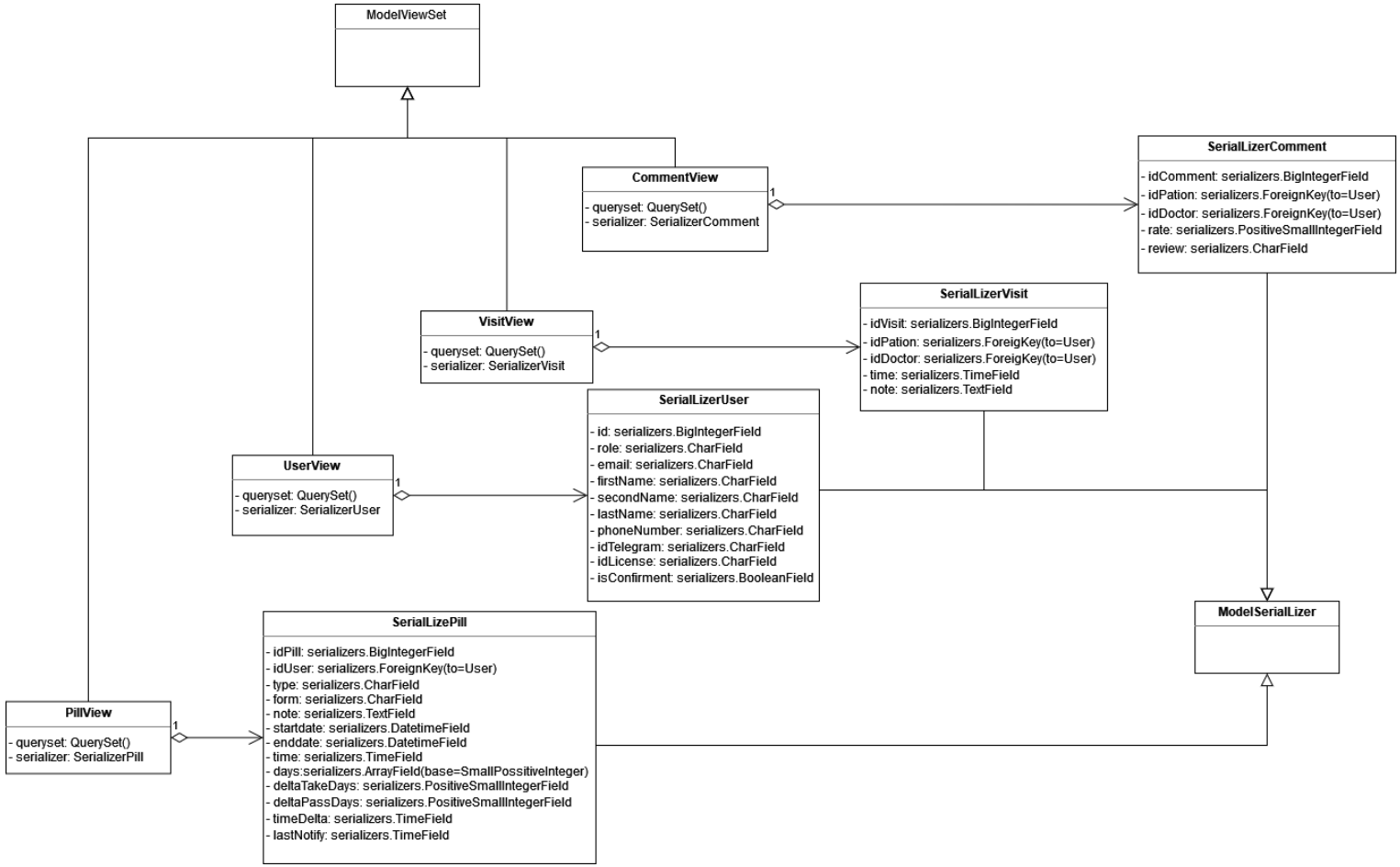

Рисунок 7.4 — Общая диаграмма классов.

На данной диаграмме можно заметить, что между соответствующими классами представления и классами сериализаторов реализована агрегация. Это необходимо, потому что это неотъемлемая часть представления, показывающая какой класс сериализатора использовать.

Таким образом была создана иерархия классов для данного проекта. Основываясь на ней в будущем будет возможность удобно добавлять новые функции по мере роста проекта. Также на данном этапе иерархия поможет непосредственно в создании проекта и в реализации общения с базой данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Аудитория интернета в 2022 году.: MediaScope [Электронный ресурс]. - 2023. - URL: https://mediascope.net/upload/iblock/3d8/qrlhud7t7dxyzw1rhtzxg3rwk8deg7uk/2022_РЧРЊРЂРХРrРЊРХРЂ.pdf (дата обращения 10.02.2023)

2. diagrams.net.: Официальный сайт. - URL: https://vk.com/wall-132685896_980 (дата обращения 10.02.2023)

3. pgAdmin.org: Официальный сайт. - URL: https://www.pgadmin.org (дата обращения 10.02.2023)

4. Python.: Официальный сайт. - URL: https://www.python.org (дата обращения 11.02.2023)

5. Django.: Официальный сайт. - URL: https://www.djangoproject.com (дата обращения 10.02.2023)

6. JavaScript.: Официальный сайт. - URL: https://developer.mozilla.org/ru/docs/Web/JavaScript (дата обращения 10.02.2023)

7. React.: Официальный сайт. - URL: https://ru.legacy.reactjs.org (дата обращения 10.02.2023)

8. Telegram.: Официальный сайт. - URL: https://telegram.org (дата обращения 10.02.2023)

9. aiogram.: Официальный сайт. - URL: https://aiogram.dev (дата обращения 10.02.2023)

10. Docker.: Официальный сайт. - URL: https://www.docker.com (дата обращения 10.02.2023)

11. Docker.: Официальный сайт. - URL: https://docs.docker.com/compose/ (дата обращения 10.02.2023)