[Орган управления образованием субъекта Российской Федерации]
[Название колледжа]

Отделение информационных технологий

ДИПЛОМНЫЙ ПРОЕКТ

Пояснительная записка

на тему: «Разработка веб-приложения для онлайн-записи пациентов в частную медицинскую клинику»

Специальность 09.02.07 «Информационные системы и программирование»

Квалификация: разработчик веб и мультимедийных приложений

Выполнил(а): студент(ка) гр. ____

_____ / И. О. Фамилия

Руководитель:

_____ / И. О. Фамилия

Консультант:

_____ / И. О. Фамилия

[Город] 2026

Пример работы, выполненной в Универ AI · univerai.ru

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....3
1. Анализ предметной области и требований.....6
1.1 Характеристика процесса записи пациентов в частную клинику.....6
1.2 Обзор существующих решений для онлайн-записи.....9
1.3 Правовые требования к обработке персональных данных пациентов.11
1.4 Функциональные и нефункциональные требования.....13
2. Проектирование веб-приложения.....17
2.1 Выбор архитектуры и технологического стека.....17
2.2 Проектирование базы данных.....20
2.3 Проектирование алгоритма записи и пользовательского интерфейса.24
2.4 Проектирование защиты персональных данных.....27
3. Реализация и тестирование.....29
3.1 Структура проекта и модели данных.....29
3.2 Реализация расчёта расписания и записи на приём.....31
3.3 Реализация личного кабинета и рабочих мест сотрудников.....33
3.4 Тестирование веб-приложения.....35
3.5 Развёртывание и сопровождение.....38
4. Экономическое обоснование проекта.....41
4.1 Расчёт затрат на разработку.....41
4.2 Затраты на внедрение и эксплуатацию.....43
4.3 Оценка экономического эффекта.....44
ЗАКЛЮЧЕНИЕ.....47
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....49
Приложение А. Скрипт создания основных таблиц базы данных.....55
Приложение Б. Выдержки из технического задания.....58

ВВЕДЕНИЕ

Частная медицина в России растёт, но уже не за счёт притока новых пациентов, а за счёт цен. По оценке консалтинговой компании Eqiva, которую в конце 2025 года привёл «Коммерсантъ FM», рынок платных медицинских услуг достиг примерно 1,78 трлн руб., однако в сопоставимых ценах прибавил лишь 5,3 %, тогда как медицинская инфляция составила 14,4 % [22]. Когда рынок перестаёт расширяться, клиники начинают бороться за каждого пациента, и удобство записи на приём превращается из мелочи в конкурентное преимущество.

Во многих небольших клиниках запись по-прежнему ведётся по телефону. Администратор держит в голове расписание десятка врачей, сверяется с таблицей или журналом и параллельно принимает посетителей у стойки. Звонки в часы пик теряются, пациенты, не дозвонившись, уходят к конкурентам, а забытые визиты оставляют врачей без работы. Это прямые потери. Веб-приложение для онлайн-записи снимает с администратора рутинную часть этой работы: пациент сам видит свободное время, сам выбирает врача и получает напоминание.

У такой системы есть особенность, которая отличает её от обычного интернет-магазина. Уже сам факт обращения человека за медицинской помощью закон относит к врачебной тайне [3], а сведения о здоровье являются специальной категорией персональных данных [2]. Поэтому разработчик обязан с первых шагов проектировать хранение, доступ и согласия так, чтобы клиника могла выполнить требования законодательства, а не дописывать защиту после запуска.

Цель дипломного проекта — разработать веб-приложение для онлайн-записи пациентов в частную медицинскую клинику «[название клиники]», которое сократит нагрузку на администраторов, исключит двойную запись на одно время и будет соответствовать требованиям к обработке персональных данных пациентов.

Для достижения цели поставлены следующие задачи:

1) проанализировать существующий порядок записи пациентов и выявить его узкие места;

2) сравнить классы готовых решений для онлайн-записи и обосновать разработку собственного приложения;

3) изучить правовые требования к обработке персональных данных пациентов;

4) сформулировать функциональные и нефункциональные требования, построить диаграмму вариантов использования;

5) спроектировать архитектуру, базу данных, алгоритм записи и пользовательский интерфейс;

6) реализовать приложение и провести его тестирование;

7) рассчитать затраты на разработку и внедрение и оценить экономический эффект.

Объект проектирования — процесс записи пациентов на приём в частной многопрофильной клинике. **Предмет** — веб-приложение, автоматизирующее этот процесс.

Методологической базой послужили стандарты на техническое задание для автоматизированных систем [8] и функциональное моделирование IDEF0 [9], руководства по моделированию на UML [10, 19] и проектированию баз данных [11, 12], документация фреймворка Django и СУБД PostgreSQL. Правовую основу составили Федеральный закон «О персональных данных», Федеральный закон «Об основах охраны здоровья граждан в Российской Федерации» и подзаконные акты о защите информации.

Практическая значимость проекта состоит в том, что приложение готово к развёртыванию в небольшой клинике на одном виртуальном сервере и не требует покупки лицензий. Проект выполнен в соответствии с требованиями ФГОС СПО по специальности 09.02.07 «Информационные системы и программирование» [7] и затрагивает компетенции сразу нескольких профессиональных модулей: проектирование, разработку, тестирование и сопровождение информационных систем.

Пояснительная записка состоит из введения, четырёх глав, заключения, списка использованных источников и двух приложений. Сведения о клинике, её загрузке и затратах не относятся к реальной организации: это условные данные для примера, они отмечены в тексте и в примечаниях к таблицам.

1. Анализ предметной области и требований

1.1 Характеристика процесса записи пациентов в частную клинику

Клиника «[название клиники]» — небольшая многопрофильная медицинская организация. Для примера примем, что в ней ведут приём девять врачей семи специальностей: терапевт, педиатр, кардиолог, невролог, гинеколог, оториноларинголог и врач ультразвуковой диагностики. Клиника работает с 8:00 до 20:00 в будни и с 9:00 до 15:00 в субботу. За стойкой регистратуры по сменам работают два администратора (здесь и далее в подразделе — условные данные для примера).

Сейчас пациент может записаться тремя путями: позвонить, написать в мессенджер или прийти лично. Расписание врачей хранится в общей электронной таблице, где каждый лист соответствует неделе, а строки — получасовым интервалам. Администратор, принимая звонок, открывает нужный лист, ищет свободную ячейку, вписывает фамилию и телефон пациента, а затем вручную переносит запись в медицинскую информационную систему, если пациент первичный.

Чтобы увидеть процесс целиком, он описан в нотации IDEF0. В этой методологии каждая функция изображается блоком, к которому подходят четыре вида стрелок: вход (что преобразуется), управление (правила и ограничения), выход (результат) и механизм (кто или что выполняет работу) [9]. Для пояснительной записки контекстная диаграмма и её декомпозиция сведены в одну схему, показанную на рисунке 1.

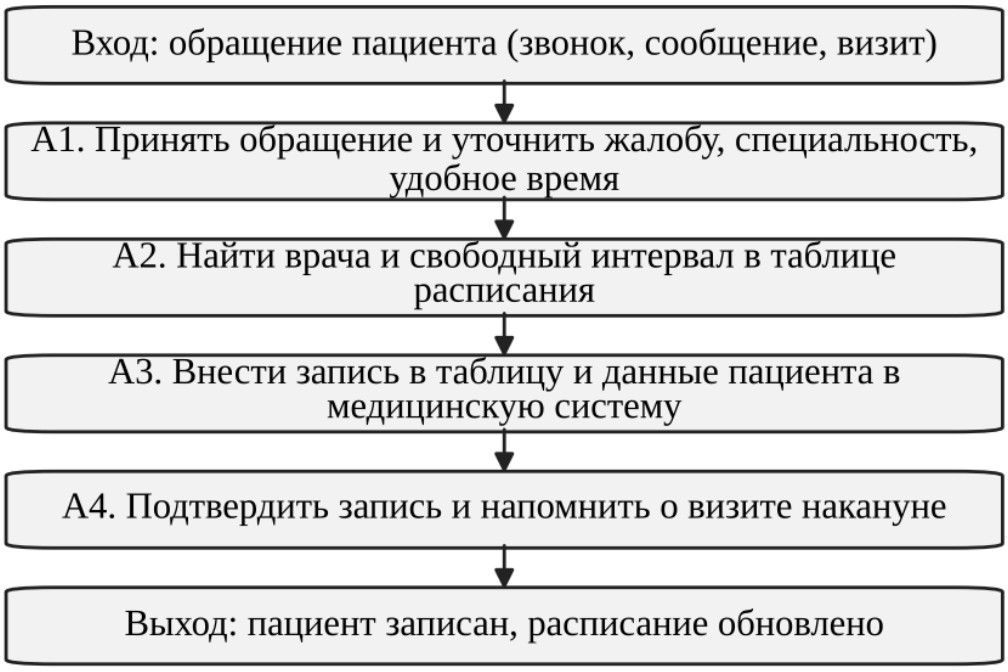


Рисунок 1 – Декомпозиция функции «Записать пациента на приём» (по методологии IDEF0, текущий процесс)

Управление: прейскурант, график работы врачей, правила записи, требования 152-ФЗ. Механизмы: администратор, телефон, электронная таблица, медицинская информационная система. Составлено автором.

Узкие места видны уже на схеме. Их три. Функции A2 и A3 выполняются вручную и в двух местах, поэтому расписание и медицинская система расходятся. Функция A4 зависит от памяти администратора: вечером, когда поток посетителей самый плотный, обзвон на следующий день часто откладывается. A1 и A2 привязаны к часам работы регистратуры, так что вечером и в воскресенье записаться невозможно.

Для оценки масштаба проблемы в течение двух недель велся журнал обращений. По его итогам в среднем за рабочий день поступает около 140 обращений, из них примерно 95 связаны с записью, переносом или отменой визита. Один такой разговор занимает у администратора от двух до четырёх минут. Набегают часы. Около 12 % пациентов не приходят на приём и не предупреждают об этом, причём больше половины неявок приходится на визиты, о которых не напомнили накануне (условные данные для примера).

Таблица 1 – Проблемы текущего процесса записи и их последствия

Проблема	**Где возникает**	**Последствие для клиники**
Запись возможна только в часы работы регистратуры	А1	Потеря пациентов, которые хотят записаться вечером или в выходной
Занятая линия в часы пик	А1	Пропущенные звонки, повторные попытки дозвониться
Расписание ведётся в таблице без блокировок	А2, А3	Двойная запись на одно время, конфликты у кабинета
Двойной ввод данных пациента	А3	Ошибки в телефонах и фамилиях, лишние минуты на каждый звонок
Ручной обзвон накануне	А4	Неявки, простой врачей
Данные пациентов в общей таблице без разграничения доступа	А3	Риск утечки персональных данных и нарушения врачебной тайны

Примечание — составлено автором по результатам анализа процесса; клиника условная.

Отдельного внимания заслуживает последняя строка таблицы. Она касается закона. Общая таблица расписания доступна всем сотрудникам, включая стажёров и внешнего бухгалтера, а в ней рядом с фамилией пациента часто записана жалоба («болит ухо», «повторно по анализам»). С точки зрения закона это уже сведения о состоянии здоровья, и хранить их в файле без разграничения доступа нельзя. Новая система должна не просто ускорить запись, но и навести порядок в том, кто и что видит.

Из анализа следует, что автоматизации прежде всего требуют функции А2–А4: поиск свободного времени, фиксация записи без конфликтов и напоминания. Функция А1 частично сохраняется за администратором, потому что пожилые пациенты будут и дальше звонить по телефону. Но и в этом случае администратор будет работать в том же приложении, что и пациенты, и видеть одно и то же расписание.

Целевой процесс после внедрения выглядит иначе. Пациент, который пользуется сайтом, проходит функции А1–А3 самостоятельно, причём в любое время суток: выбирает врача, видит только действительно свободные интервалы и сразу получает подтверждение. Для звонящих пациентов администратор выполняет те же шаги, но в том же приложении, поэтому двойного ввода больше нет. Функция А4 полностью передаётся системе:

напоминание уходит автоматически за сутки до визита, а ответ «не смогу прийти» освобождает интервал для других. Управляющие воздействия при этом не меняются, меняется механизм: вместо таблицы и памяти администратора работают база данных и правила, заложенные в код.

1.2 Обзор существующих решений для онлайн-записи

Прежде чем писать собственное приложение, нужно убедиться, что готовые решения не закрывают задачу. Сервисы онлайн-записи к врачу на российском рынке удобно разделить на четыре класса: медицинские агрегаторы с каталогом врачей и отзывами, модули записи в составе медицинских информационных систем (МИС), универсальные конструкторы онлайн-записи для сферы услуг и собственная разработка под конкретную клинику.

Агрегаторы привлекают новых пациентов и берут на себя маркетинг, но клиника платит за каждую запись или за размещение, а пациент записывается на чужой площадке и знакомится в первую очередь с брендом агрегатора. Модули МИС глубоко интегрированы с медицинской картой, но рассчитаны на клиники, которые уже купили и внедрили эту систему. Для маленькой организации без МИС их стоимость с лицензиями и внедрением оказывается непропорциональной. Универсальные конструкторы дешевы и быстро настраиваются, однако создаются для салонов, автосервисов и фитнес-клубов, поэтому медицинская специфика (направления, услуги с разной длительностью, врачебная тайна) в них учтена слабо.

Таблица 2 – Сравнение классов решений для онлайн-записи

Критерий	**Агрегатор**	**Модуль МИС**	**Конструктор записи**	**Собственное приложение**
Запись на сайте клиники под её брендом	Нет, на площадке агрегатора	Да, через виджет	Да, через виджет	Да
Учёт длительности разных услуг	Частично	Да	Да	Да
Зависимость от тарифов поставщика	Высокая, оплата за запись	Высокая, лицензии	Средняя, подписка	Нет, только хостинг
Размещение данных пациентов	У агрегатора	У поставщика или в клинике	У поставщика	На сервере, выбранном клиникой
Доработка под процессы клиники	Нет	Ограниченно, силами поставщика	Нет	Да
Стоимость первичного внедрения	Низкая	Высокая	Низкая	Средняя

Примечание — составлено автором; оценки обобщённые, для конкретного продукта их нужно уточнять по договору и тарифам поставщика.

Для клиники «[название клиники]» важны три обстоятельства. Во-первых, медицинской информационной системы с модулем записи у неё нет, а внедрять её только ради расписания экономически невыгодно. Во-вторых, руководство хочет, чтобы запись велась на собственном сайте и данные пациентов не уходили третьим лицам без необходимости: при передаче обработки другому лицу закон требует отдельного поручения и согласия пациента на такую передачу [2]. В-третьих, клинике нужна гибкость в правилах, например запрет повторной записи к одному врачу на один день и разная длительность первичного и повторного приёма.

Собственное приложение не исключает работы с агрегаторами. Если позднее клиника захочет размещаться на внешних площадках, приложение может стать для них источником расписания через программный интерфейс. Но ядро — расписание, записи и данные пациентов — остаётся под контролем клиники. Этот вывод и определил постановку задачи на разработку.

1.3 Правовые требования к обработке персональных данных пациентов

Главный закон, с которым работает система записи, — Федеральный закон от 27.07.2006 № 152-ФЗ «О персональных данных». Клиника выступает в нём оператором, то есть организует обработку и определяет её цели. Фамилия, имя, телефон и адрес электронной почты пациента — обычные персональные данные. Но статья 10 закона выделяет специальные категории, к которым относятся сведения о состоянии здоровья, и запрещает их обработку, кроме прямо перечисленных случаев [2].

Для клиники значимы два исключения из этого запрета. Оба узкие. Первое — письменное согласие субъекта (п. 1 ч. 2 ст. 10). Второе — обработка в медико-профилактических целях, для установления диагноза и оказания медицинских услуг, если её ведёт лицо, профессионально занимающееся медицинской деятельностью и обязанное хранить врачебную тайну (п. 4 ч. 2 ст. 10) [2]. На этапе онлайн-записи, когда пациент ещё не заключил договор и не пришёл на приём, безопаснее опираться на согласие, а сведения о жалобах вообще не собирать без необходимости.

Статья 13 Федерального закона от 21.11.2011 № 323-ФЗ относит к врачебной тайне не только диагноз, но и сведения о факте обращения гражданина за медицинской помощью [3]. Отсюда практический вывод для проектирования: даже запись «Иванова И. И., 14:30, гинеколог» без единого слова о жалобах уже является охраняемой информацией. Значит, список записей нельзя показывать никому, кроме сотрудников, которым он нужен по работе, а уведомления не должны раскрывать специальность врача постороннему, например на экране заблокированного телефона.

Требования к самому согласию заметно ужесточились. Особенно в 2025 году. Согласие должно быть конкретным, предметным, информированным, сознательным и однозначным, а часть 4 статьи 9 перечисляет его обязательные реквизиты: сведения о субъекте и операторе, цель обработки, перечень данных, перечень действий, срок действия и

порядок отзыва [2]. С 1 сентября 2025 года, после изменений, внесённых Федеральным законом от 24.06.2025 № 156-ФЗ, согласие должно оформляться отдельно от иных документов, которые подтверждает или подписывает субъект [2, 23]. Для веб-формы это означает отдельный флажок и отдельную страницу с текстом согласия, а не строчку мелким шрифтом внутри пользовательского соглашения.

Ещё несколько норм закона прямо влияют на архитектуру. Перечислим их. Оператор, собирающий данные через интернет, обязан опубликовать на сайте свою политику обработки персональных данных (ч. 2 ст. 18.1). Запись, систематизация, хранение и извлечение персональных данных граждан России с использованием баз данных за пределами страны не допускаются (ч. 5 ст. 18), поэтому сервер приложения и резервные копии должны располагаться в российском центре обработки данных. До начала обработки оператор обязан уведомить Роскомнадзор (ч. 1 ст. 22), а при утечке — сообщить о ней в течение 24 часов и о результатах внутреннего расследования в течение 72 часов (ч. 3.1 ст. 21) [2]. Чтобы клиника могла расследовать инцидент, приложение должно вести журнал входов и действий с записями.

Таблица 3 – Требования законодательства и их отражение в проекте

Норма	**Содержание требования**	**Решение в приложении**
152-ФЗ, ст. 9	Отдельное, конкретное и информированное согласие	Отдельная страница согласия, флажок без предварительной отметки, хранение версии текста и времени согласия
152-ФЗ, ст. 10; 323-ФЗ, ст. 13	Здоровье и факт обращения — охраняемые сведения	Поле жалобы необязательное, ролевой доступ, нейтральный текст уведомлений
152-ФЗ, ч. 5 ст. 18	Базы данных — на территории России	Хостинг и резервные копии в российском ЦОД
152-ФЗ, ч. 2 ст. 18.1	Публикация политики обработки	Страница политики, ссылка в подвале и в форме записи
152-ФЗ, ч. 3.1 ст. 21	Уведомление об инциденте за 24 и 72 часа	Журнал аудита действий пользователей
Постановление Правительства РФ № 1119	Уровень защищённости информационной системы	Меры для третьего уровня (подраздел 2.4)

Примечание — составлено автором по [2, 3, 4].

Конкретный набор мер защиты зависит от уровня защищённости информационной системы, который определяется по постановлению Правительства РФ от 01.11.2012 № 1119. Система клиники обрабатывает специальные категории данных менее чем 100 тысяч субъектов, не являющихся её сотрудниками. Если для неё актуальны угрозы третьего типа (то есть не связанные с недокументированными возможностями системного и прикладного ПО), требуется третий уровень защищённости (подп. «в» п. 11) [4]. Состав организационных и технических мер для каждого уровня установлен приказом ФСТЭК России от 18.02.2013 № 21 [6]; часть из них, например идентификация и аутентификация пользователей, управление доступом и регистрация событий безопасности, реализуется средствами самого приложения.

Наконец, Правила предоставления медицинскими организациями платных медицинских услуг, утверждённые постановлением Правительства РФ от 11.05.2023 № 736 и действующие с 1 сентября 2023 года, обязывают исполнителя доводить до потребителя перечень платных услуг с ценами, сведения о медицинских работниках и график их работы (п. 17) [5]. Каталог врачей с образованием и квалификацией, прейскурант и открытое расписание в приложении помогают клинике выполнить и это требование.

1.4 Функциональные и нефункциональные требования

Требования собраны в логике вариантов использования, которую описывают руководства по UML: сначала выделены классы пользователей и их цели, затем варианты использования, и только после этого — конкретные функции и ограничения [19]. Источниками послужили беседы с условными администраторами и врачами клиники, анализ процесса из подраздела 1.1 и правовые требования из подраздела 1.3. Структура итогового документа следует разделам технического задания по ГОСТ 34.602-2020 [8].

В системе выделены четыре роли. У каждой свои цели. Пациент записывается, переносит и отменяет визиты, видит свою историю записей.

Администратор ведёт расписание, записывает пациентов по телефону и отмечает явку. Врач просматривает свой список пациентов на день. Руководитель клиники настраивает справочники (специальности, услуги, кабинеты) и смотрит отчёты. Роли и их варианты использования показаны на рисунке 2 в виде иерархии, которая заменяет классическую диаграмму вариантов использования UML [10, 13, 14].

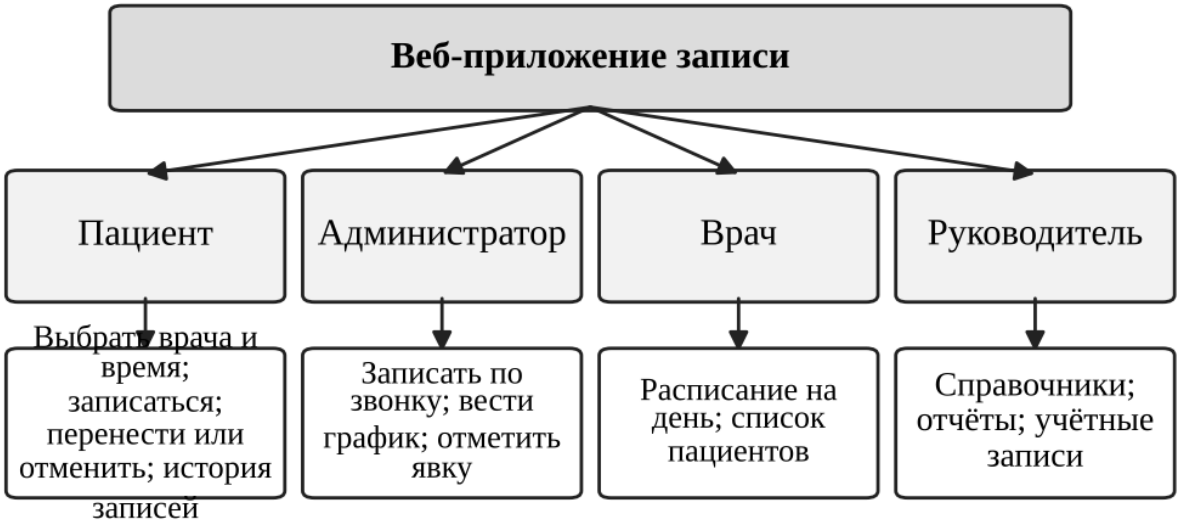


Рисунок 2 – Варианты использования веб-приложения по ролям пользователей

Составлено автором.

Подробно описан основной вариант использования «Записаться на приём». Предусловие: пациент открыл сайт клиники. Основной поток: пациент выбирает специальность или врача, затем услугу; система показывает ближайшие свободные интервалы с учётом длительности услуги; пациент выбирает интервал, вводит фамилию, имя, телефон и адрес почты, отмечает согласие на обработку персональных данных; система повторно проверяет, что интервал свободен, создаёт запись и отправляет подтверждение. Альтернативный поток: если за время заполнения формы интервал занял другой пациент, система сообщает об этом и предлагает соседние интервалы. Постусловие: запись создана в статусе «запланирована», интервал недоступен другим пациентам.

Таблица 4 – Функциональные требования к веб-приложению

Код	Требование	Роль	Приоритет
ФТ-1	Просмотр каталога врачей с образованием, квалификацией и услугами	Пациент	Высокий
ФТ-2	Показ свободных интервалов с учётом длительности услуги	Пациент, администратор	Высокий
ФТ-3	Создание записи с проверкой занятости интервала в момент сохранения	Пациент, администратор	Высокий
ФТ-4	Получение отдельного согласия на обработку персональных данных	Пациент	Высокий
ФТ-5	Подтверждение записи по электронной почте и напоминание за сутки	Система	Высокий
ФТ-6	Перенос и отмена записи не позднее чем за 2 часа до приёма	Пациент	Средний
ФТ-7	Ведение графика работы врачей и исключений (отпуск, болезнь)	Администратор	Высокий
ФТ-8	Отметка явки и неявки пациента	Администратор	Средний
ФТ-9	Список пациентов врача на выбранный день	Врач	Средний
ФТ-10	Справочники специальностей, услуг, кабинетов	Руководитель	Средний
ФТ-11	Отчёт о загрузке врачей и доле неявок за период	Руководитель	Низкий
ФТ-12	Журнал действий пользователей с персональными данными	Система	Высокий

Примечание — составлено автором.

Приоритеты в таблице расставлены вместе с условным руководителем клиники. Высокий приоритет получили требования, без которых приложение нельзя запускать: запись без конфликтов, согласие, уведомления, график врачей и журнал аудита. Средний — функции, которые экономят время, но могут подождать первой доработки. Низкий приоритет у единственного требования, отчёта о загрузке: до запуска приложения считать в нём просто нечего. Такое деление позволило выпустить первую рабочую версию раньше, а остальное добавить по ходу опытной эксплуатации.

Нефункциональные требования сгруппированы по характеристикам качества. Производительность: страница выбора времени должна открываться не дольше чем за 1 секунду при 50 одновременных пользователях. Надёжность: резервная копия базы данных создаётся ежедневно и хранится 30 дней. Безопасность: пароли хранятся только в виде хеша, все страницы отдаются по HTTPS, после пяти неудачных попыток

входа учётная запись временно блокируется. Удобство: форма записи занимает не более трёх шагов и корректно отображается на экране смартфона шириной от 360 пикселей. Доступность: контрастность текста и подписи полей должны соответствовать уровню AA рекомендаций WCAG 2.1, потому что среди пациентов клиники много пожилых людей [40].

Особое требование касается времени. Точнее, часовых поясов. Клиника работает в одном часовом поясе, но сервер может быть настроен на всемирное координированное время. Чтобы запись на 9:00 не превратилась в запись на 6:00 или 12:00, все моменты времени хранятся в базе с часовым поясом, а пересчёт в местное время выполняется только при выводе на экран.

Сформулированные требования стали основой технического задания, выдержки из которого приведены в приложении Б. В следующей главе на их основе выбираются архитектура и технологии, проектируются база данных и интерфейс.

2. Проектирование веб-приложения

2.1 Выбор архитектуры и технологического стека

Приложение записи — типичная информационная система с веб-интерфейсом: несколько видов пользователей, справочники, транзакционная таблица записей и отчёты. Для такого класса задач М. Фаулер рекомендует разделять представление, предметную логику и источник данных, чтобы изменение интерфейса не затрагивало правила записи, а смена способа хранения не ломала интерфейс [18]. Р. Мартин формулирует ту же мысль жёстче: бизнес-правила не должны зависеть от фреймворка, базы данных и пользовательского интерфейса, а зависимости в исходном коде направлены только внутрь, к правилам [31].

Для небольшой клиники полностью «чистая» архитектура с отдельными слоями интерфейсов и адаптеров избыточна. Выбран компромисс: классическая трёхзвенная схема (браузер — сервер приложения — СУБД), но с выделенным модулем сервисов, где сосредоточены правила записи. Представления (контроллеры) только принимают запрос, вызывают сервис и отдают ответ. Такую логику можно проверить модульными тестами без запуска веб-сервера.

Рассматривался и вариант с микросервисами, когда расписание, записи и уведомления живут в отдельных программах и общаются по сети. Для крупной сети клиник это оправдано: команды работают независимо, а нагруженную часть можно масштабировать отдельно. Для одной клиники с девятью врачами такая схема дала бы только издержки. Пришлось бы согласовывать данные между сервисами, а сбой сети между ними стал бы новой причиной двойной записи. Поэтому приложение собрано как модульный монолит: один процесс, одна база данных, но чёткие границы между модулями. Если клиника вырастет, модуль уведомлений выделяется в отдельный сервис первым, потому что он уже сейчас общается с остальной системой только через таблицу очереди.

Выбор языка и фреймворка проводился среди четырёх распространённых вариантов. Оценивались встроенные средства, которые иначе пришлось бы писать вручную (панель администратора, аутентификация, миграции базы данных), зрелость средств защиты, стоимость хостинга и знакомство команды с технологией. Результат сведён в таблицу 5.

Таблица 5 – Сравнение технологических стеков для серверной части

Критерий	**Python / Django**	**PHP / Laravel**	**JavaScript / Node.js + Express**	**C# / ASP.NET Core**
Готовая панель администратора	Встроенная	Отдельный пакет	Нет	Нет
ORM и миграции из коробки	Да	Да	Отдельные библиотеки	Да (Entity Framework)
Встроенная защита от CSRF, XSS, SQL-инъекций	Да	Да	Частично, через пакеты	Да
Версия с долгосрочной поддержкой	Да (LTS)	Нет, короткий цикл	Да для Node.js	Да (LTS)
Требования к серверу	Linux, низкие	Linux, низкие	Linux, низкие	Linux, средние
Опыт разработчика	Высокий	Средний	Средний	Низкий

Примечание — составлено автором; строка «Опыт разработчика» отражает навыки автора проекта.

Выбор сделан в пользу Python и Django. Причин две. Решающими оказались встроенная панель администратора, которая закрывает почти все потребности руководителя клиники в справочниках без отдельной разработки, и встроенные механизмы безопасности. Django по умолчанию защищает формы от подделки межсайтовых запросов, экранирует данные в шаблонах и формирует параметризованные SQL-запросы через ORM [29]. Для проекта взята ветка Django 5.2 с долгосрочной поддержкой: исправления безопасности для неё выпускаются как минимум до апреля 2028 года, а Python 3.12 входит в число поддерживаемых ею версий языка [25].

В качестве СУБД выбрана PostgreSQL 16. Помимо надёжной транзакционной модели [17] в ней есть два механизма, которые прямо

решают главную проблему старого процесса — двойную запись. Частичный индекс позволяет сделать уникальной пару «врач — время начала» только для активных записей, не мешая хранить отменённые [36]. Ограничение-исключение запрещает пересечение интервалов времени у одного врача, даже если услуги имеют разную длительность [35]. В MySQL и SQLite таких средств нет или они реализуются триггерами. Это решило выбор.

Интерфейс строится на серверных шаблонах Django с небольшой долей JavaScript, а не в виде одностраничного приложения. Пациенту нужно заполнить три экрана, и сложный клиентский фреймворк добавил бы сборку, отдельный программный интерфейс и удвоенную проверку данных, не дав заметной выгоды. Адаптивная вёрстка выполнена на Bootstrap 5, сетка которого рассчитана на экраны от смартфона до настольного монитора [24]. Свободные интервалы подгружаются запросом GET к адресу, который возвращает JSON. По спецификации HTTP метод GET является безопасным и не должен менять состояние сервера, поэтому создание записи выполняется только методом POST [21, 37].

Настройки (пароль к базе данных, ключ шифрования сессий, параметры почтового сервера) хранятся в переменных окружения, а не в исходном коде, как советует методология The Twelve-Factor App [39]. Это позволяет держать код в открытом репозитории без риска раскрыть пароли и разворачивать одну и ту же сборку на тестовом и рабочем серверах. Общая схема развёртывания приведена на рисунке 3.

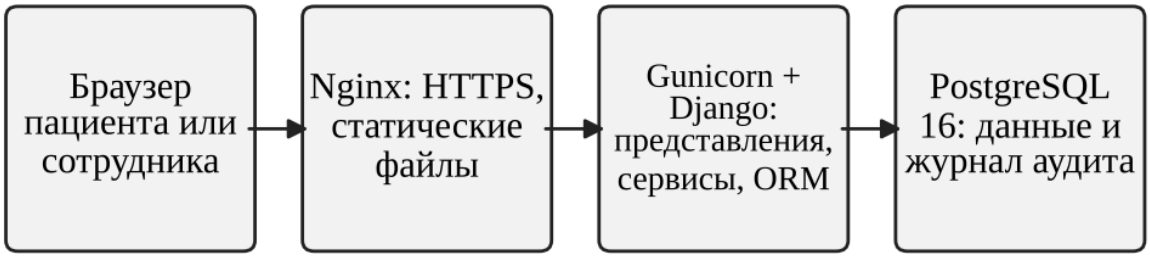


Рисунок 3 – Архитектура развёртывания веб-приложения

Отдельно по расписанию cron запускается команда отправки напоминаний, которая обращается к той же базе и к почтовому серверу. Составлено автором.

Все компоненты размещаются на одном виртуальном сервере в российском центре обработки данных, что выполняет требование ч. 5 ст. 18 Закона о персональных данных [2]. Резервные копии базы данных шифруются и копируются в объектное хранилище того же провайдера. При росте нагрузки СУБД можно вынести на отдельный сервер без изменения кода: достаточно поменять переменную окружения с адресом базы данных.

2.2 Проектирование базы данных

Модель данных построена в два этапа. Сначала была составлена концептуальная модель «сущность — связь», предложенная П. Ченом: предметная область описывается через сущности, их атрибуты и связи с указанием кратности [20]. Затем модель преобразована в реляционную схему и проверена на соответствие третьей нормальной форме, чтобы каждый неключевой атрибут зависел только от ключа и ни от чего другого [11, 12].

В результате анализа выделено одиннадцать сущностей. Их назначение, ключевые атрибуты и связи приведены в таблице 6. Учётные записи сотрудников и пациентов хранятся в стандартной таблице пользователей Django, а профиль пациента и профиль врача ссылаются на неё связью «один к одному». Так пациент может войти в личный кабинет, а врач — в своё расписание с одним и тем же механизмом аутентификации.

Таблица 6 – Сущности логической модели данных

Сущность	Основные атрибуты	Связи
Специальность	id, название, описание	1 : N с врачом
Врач	id, ФИО, образование, квалификация, стаж, фото, активен	N : 1 со специальностью; N : M с услугой; 1 : N с графиком
Услуга	id, название, длительность, цена, первичная или повторная	N : M с врачом; 1 : N с записью
Кабинет	id, номер, этаж	1 : N с графиком
График работы	id, врач, кабинет, день недели, начало, окончание	N : 1 с врачом и кабинетом
Исключение графика	id, врач, дата, тип (отпуск, болезнь, доп. смена), начало, окончание	N : 1 с врачом
Пациент	id, пользователь, фамилия, имя, отчество, телефон, e-mail, дата рождения	1 : N с записью и согласием
Запись	id, пациент, врач, услуга, начало, окончание, статус, источник, комментарий, создана	N : 1 с пациентом, врачом и услугой
Текст согласия	id, версия, текст, действует с	1 : N с согласием
Согласие	id, пациент, текст согласия, дата и время, IP-адрес, отозвано	N : 1 с пациентом и текстом согласия
Журнал аудита	id, пользователь, действие, объект, время, IP-адрес	N : 1 с пользователем

Примечание — составлено автором.

Центральная сущность модели — «Запись». Вокруг неё группируются справочники, из которых складывается один визит пациента. Её окружение показано на рисунке 4. Второй узел модели — врач с его графиком и исключениями: именно из них при каждом запросе вычисляются свободные интервалы (рисунок 5).

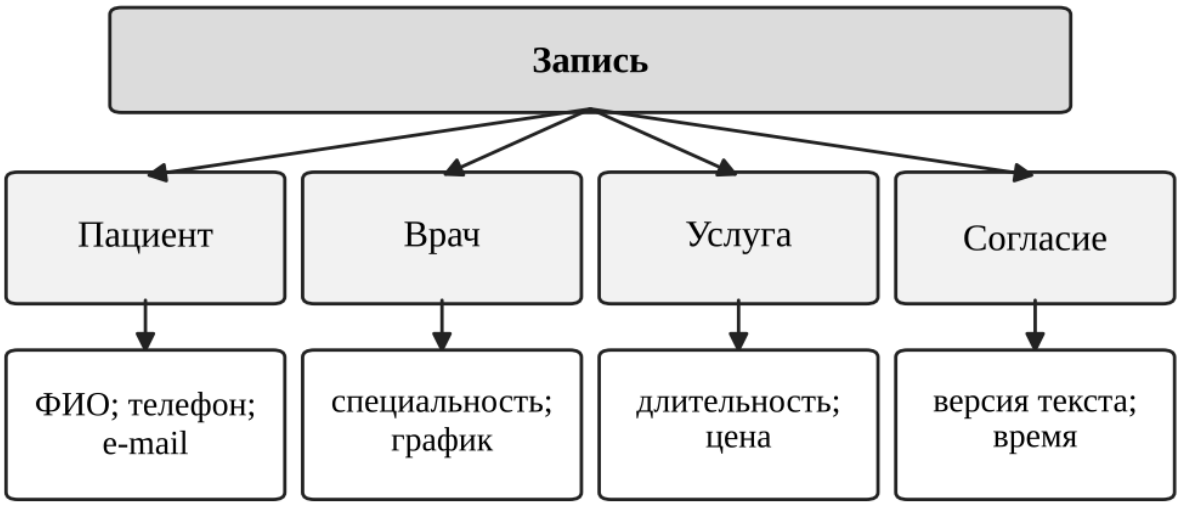


Рисунок 4 – Фрагмент ER-модели: сущность «Запись» и связанные сущности

Связи «Пациент», «Врач», «Услуга» с «Записью» — один ко многим; «Согласие» связано с пациентом и действует для всех его записей. Составлено автором.

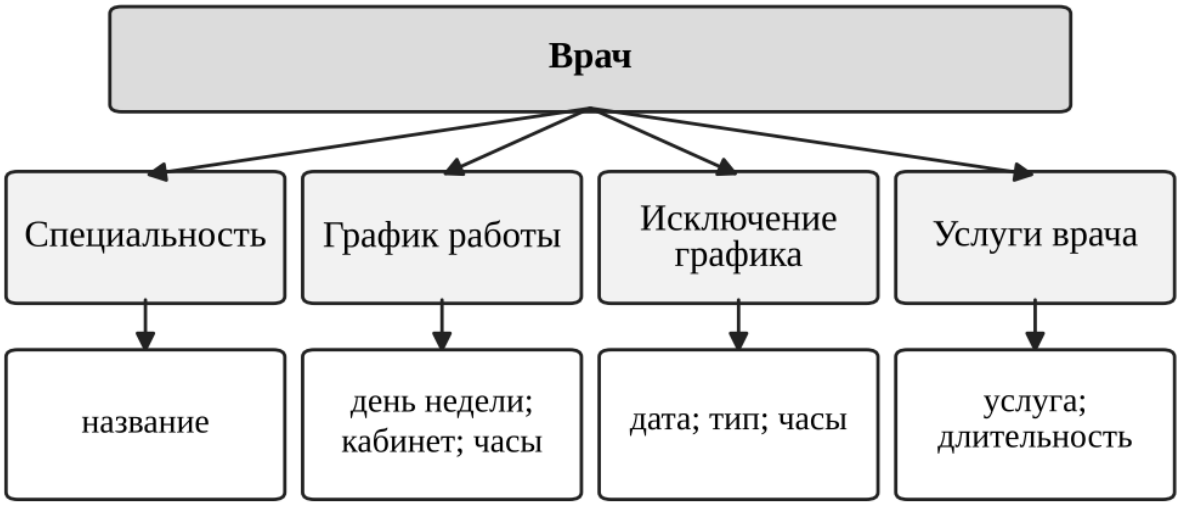


Рисунок 5 – Фрагмент ER-модели: врач, его график и исключения

Составлено автором.

Нормализация хорошо видна на примере старой таблицы расписания. В ней каждая строка содержала дату, время, фамилию врача, его специальность, номер кабинета, фамилию и телефон пациента. Специальность и кабинет повторялись в сотнях строк и зависели не от записи, а от врача и его графика, то есть таблица нарушала третью нормальную форму: неключевые атрибуты зависели от другого неключевого атрибута. Стоило врачу переехать в другой кабинет, и правку приходилось вносить во все будущие строки. В новой схеме специальность хранится один раз в карточке врача, кабинет — в строке графика, а запись ссылается только на врача, пациента и услугу.

Отдельно стоит пояснить, почему в модели нет таблицы готовых «слотов». Хранить заранее нарезанные получасовые интервалы на месяц вперёд проще для выборки, но такая таблица быстро расходится с действительностью: врач меняет график, услуги имеют разную длительность, а при каждом изменении слоты приходится перегенерировать. Поэтому в базе хранятся только правила (график и исключения) и факты (записи), а свободное время вычисляется при запросе. Для девяти врачей и горизонта

записи в 30 дней это несколько тысяч интервалов, и расчёт занимает доли секунды.

Структура таблицы записей, на которую приходится основная нагрузка, приведена в таблице 7. Время начала и окончания хранится в типе timestamp with time zone: PostgreSQL сохраняет момент в UTC и пересчитывает его в часовой пояс сеанса при выдаче [16].

Таблица 7 – Структура таблицы appointment (запись на приём)

Поле	Тип	Ограничения	Назначение
id	bigint	PK, автоинкремент	Идентификатор записи
patient_id	bigint	FK → patient, NOT NULL	Пациент
doctor_id	bigint	FK → doctor, NOT NULL	Врач
service_id	bigint	FK → service, NOT NULL	Услуга
starts_at	timestamptz	NOT NULL	Начало приёма
ends_at	timestamptz	NOT NULL, > starts_at	Окончание приёма
status	varchar(12)	planned, done, no_show, cancelled	Статус записи
source	varchar(8)	site, phone	Способ записи
comment	varchar(300)	NULL	Необязательный комментарий пациента
created_at	timestamptz	NOT NULL, по умолчанию now()	Время создания

Примечание — дополнительно: частичный уникальный индекс (doctor_id, starts_at) WHERE status <> 'cancelled' и ограничение-исключение по пересечению интервалов (doctor_id, tstzrange(starts_at, ends_at)). Составлено автором.

Целостность расписания обеспечивается на двух уровнях. Так надёжнее. На уровне приложения сервис записи проверяет свободное время и блокирует строку врача на время транзакции. На уровне базы данных ограничение-исключение с оператором пересечения диапазонов не даст сохранить вторую активную запись, пересекающуюся с первой, даже если проверка в коде по какой-то причине будет обойдена. Для работы ограничения по целочисленному полю врача вместе с диапазоном требуется расширение btree_gist [26, 35]. Полный скрипт создания таблиц приведён в приложении А.

Поле комментария сделано необязательным и ограничено 300 символами. На форме рядом с ним написано, что описывать симптомы не нужно и всё необходимое врач уточнит на приёме. Это прямое следствие

принципа минимизации: обрабатываемые данные не должны быть избыточными по отношению к целям обработки [2].

2.3 Проектирование алгоритма записи и пользовательского интерфейса

Алгоритм записи состоит из двух частей: расчёта свободных интервалов и сохранения выбранного интервала. Расчёт выполняется для выбранного врача, услуги и даты. Сначала берётся недельный график врача на этот день недели, затем к нему применяются исключения на конкретную дату: отпуск или болезнь убирают рабочие часы, дополнительная смена добавляет. Получившееся рабочее окно делится на интервалы с шагом 15 минут, и для каждого проверяется, что интервал длиной в услугу целиком помещается в окно и не пересекается с уже существующими активными записями. Интервалы в прошлом и ближе чем через час от текущего момента отбрасываются.

Сохранение устроено сложнее. Причина в конкуренции пользователей: между показом свободного времени и нажатием кнопки «Записаться» может пройти несколько минут, за которые тот же интервал выберет другой пациент. Классическое решение — пессимистическая блокировка: транзакция блокирует строку врача командой SELECT … FOR UPDATE, повторно проверяет занятость и только затем вставляет запись [28]. Вторая транзакция, пришедшая за тем же врачом, ждёт окончания первой и после этого видит уже занятый интервал. Последовательность шагов показана на рисунке 6.

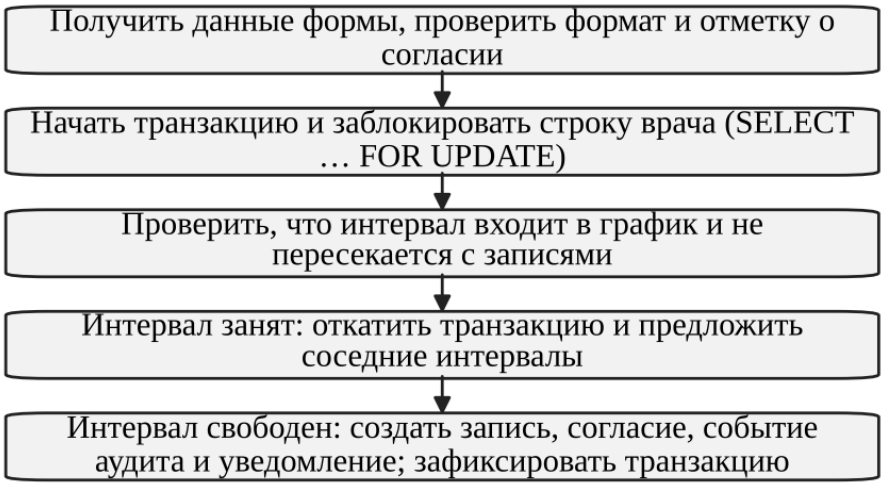


Рисунок 6 – Алгоритм сохранения записи на приём

Шаг 4 выполняется вместо шага 5, если проверка на шаге 3 не пройдена. Составлено автором.

Интерфейс пациента проектировался с оглядкой на эвристики юзабилити Я. Нильсена [32]. Система всё время показывает, на каком шаге находится пользователь (видимость состояния): над формой выводится строка «Врач → Время → Данные». Язык интерфейса — язык пациента, а не регистратуры: вместо «слот» пишется «время приёма», вместо «первичный/повторный» — «впервые у этого врача» и «повторный приём». Ошибки предупреждаются, а не исправляются: телефон вводится по маске, а недоступные даты в календаре просто нельзя нажать.

Расписание регистратуры сделано в виде сетки, где столбцы — врачи, а строки — время. Администратору важно одним взглядом видеть, у кого есть окно в ближайший час, поэтому свободные интервалы выделены светлым фоном, а записи с сайта отличаются от записей по телефону значком. Пожилым пациентам, которые звонят по телефону, администратор оформляет запись в том же интерфейсе; согласие на обработку данных у них берётся в письменной форме при первом визите, а в записи ставится отметка о способе получения.

Вёрстка проектировалась сначала для смартфона, затем для настольного экрана. Кнопки и поля имеют высоту не менее 44 пикселей, подписи полей расположены над ними, а не внутри, чтобы не исчезать при

вводе. Контраст основного текста к фону выбран не ниже 4,5 : 1, что соответствует критерию 1.4.3 рекомендаций WCAG 2.1 для уровня AA [40].

Отдельно спроектированы тексты уведомлений, потому что пациент видит их чаще, чем сам сайт. Подтверждение приходит сразу после записи и содержит дату, время, адрес клиники и ссылку на личный кабинет. Напоминание отправляется за сутки и содержит две ссылки: «Приду» и «Не смогу прийти». Вторая ссылка отменяет запись без входа в кабинет: в ней передаётся одноразовый подписанный токен, срок действия которого истекает в момент начала приёма. Так пациенту проще предупредить клинику, чем промолчать, а освободившийся интервал сразу становится доступен другим. Письма отправляются с адреса клиники, а не с адреса сервиса рассылок, иначе их чаще помечают как спам.

Перечень экранных форм и их основных элементов приведён в таблице 8.

Таблица 8 – Экранные формы веб-приложения

Экран	Роль	Основные элементы
Главная и каталог врачей	Все	Поиск по специальности, карточки врачей с образованием и квалификацией, кнопка «Записаться»
Выбор услуги и времени	Пациент	Список услуг с ценой и длительностью, календарь на 30 дней, сетка свободного времени
Данные пациента	Пациент	ФИО, телефон, e-mail, необязательный комментарий, флажок согласия со ссылкой на текст
Подтверждение	Пациент	Сводка записи, адрес клиники, кнопки «Добавить в календарь» и «Отменить»
Личный кабинет	Пациент	Будущие и прошлые записи, перенос и отмена, отзыв согласия
Расписание регистратуры	Администратор	Сетка всех врачей на день, запись по звонку, отметка явки
Мой день	Врач	Список пациентов по времени, услуга, отметка «принят»
Панель управления	Руководитель	Справочники, графики, отчёт о загрузке и неявках

Примечание — составлено автором.

2.4 Проектирование защиты персональных данных

Как показано в подразделе 1.3, для системы клиники требуется третий уровень защищённости персональных данных [4]. Часть мер из приказа ФСТЭК России № 21 носит организационный характер (назначение ответственного, утверждение перечня лиц с доступом, учёт машинных носителей) и выполняется клиникой. Другую часть берёт на себя приложение, а третью — хостинг-провайдер и настройка сервера [6].

При выборе технических мер учитывался перечень наиболее распространённых рисков веб-приложений OWASP Top 10 в редакции 2021 года. Первые места в нём занимают нарушение контроля доступа (A01), криптографические сбои (A02) и внедрение кода, включая SQL-инъекции (A03) [34]. Для системы записи самый реальный сценарий — именно A01: пациент меняет номер записи в адресной строке и видит чужой визит. Поэтому каждый запрос к записи проверяет, что она принадлежит текущему пациенту, а для сотрудников — что их роль допускает просмотр.

Таблица 9 – Меры защиты персональных данных и их реализация

Группа мер (приказ ФСТЭК № 21)	**Реализация в проекте**
Идентификация и аутентификация	Вход по e-mail и паролю; для сотрудников — обязательный второй фактор (одноразовый код); блокировка после 5 неудачных попыток
Управление доступом	Роли «пациент», «администратор», «врач», «руководитель» на основе групп Django; проверка владельца записи в каждом представлении
Регистрация событий безопасности	Журнал аудита: входы, просмотр и изменение записей, выгрузка отчётов; хранение 1 год
Защита информационной системы и каналов связи	HTTPS с TLS 1.2 и выше, заголовок HSTS, защищённые cookie сессии, защита от CSRF и кликджекинга
Антивирусная защита, контроль целостности	Средствами хостинга и сервера; контроль версий зависимостей
Обеспечение доступности	Ежедневное зашифрованное резервное копирование, проверка восстановления раз в месяц

Примечание — составлено автором по [6, 29, 34]; организационные меры в таблицу не включены.

Пароли не хранятся в открытом виде. Django по умолчанию использует алгоритм PBKDF2 с хеш-функцией SHA-256 и большим числом итераций, а

также поддерживает Argon2 [27]. Рабочая группа OWASP рекомендует для новых систем именно Argon2id, а PBKDF2 — там, где требуется соответствие стандартам FIPS [33]. В проекте первым в списке хешеров указан Argon2, поэтому новые пароли сразу хешируются им, а старые прозрачно перехешируются при следующем входе.

Уведомления по электронной почте и SMS составляются так, чтобы не раскрывать врачебную тайну постороннему, увидевшему экран телефона. В тексте напоминания указаны клиника, дата и время, а фамилия врача и специальность — только после входа в личный кабинет по ссылке. Это неудобно на первый взгляд. Зато безопасно, ведь запись «завтра в 10:00 к онкологу», всплывшая на экране, раскрывает о человеке больше, чем ему хотелось бы.

Отзыв согласия реализован в личном кабинете одной кнопкой. Без звонков и письменных заявлений. По закону после отзыва оператор должен прекратить обработку, если для неё нет иных законных оснований [2]. Приложение отменяет будущие записи, обезличивает профиль пациента и сохраняет лишь то, что клиника обязана хранить в силу других норм, например сведения об уже оказанных услугах. Решение о сроках хранения таких сведений принимает клиника, приложение лишь позволяет его исполнить.

3. Реализация и тестирование

3.1 Структура проекта и модели данных

Проект создан стандартной командой django-admin startproject и разбит на пять приложений Django, каждое из которых отвечает за свою часть предметной области. Приложение catalog содержит специальности, врачей, услуги и кабинеты; schedule — графики работы, исключения и расчёт свободного времени; booking — записи, согласия и очередь уведомлений; accounts — профили пациентов и роли сотрудников; audit — журнал действий. Правила записи вынесены в модули services.py внутри приложений schedule и booking, а представления только вызывают их.

Зависимости зафиксированы в файле requirements.txt с точными версиями: Django 5.2, драйвер psycopg 3 для PostgreSQL, argon2-cffi для хеширования паролей, django-environ для чтения переменных окружения и gunicorn в качестве сервера приложения. Исходный код хранится в репозитории Git; ветка main соответствует рабочему серверу, разработка ведётся в отдельных ветках и попадает в main только после прохождения тестов.

Таблица 10 – Структура исходного кода проекта

Каталог или файл	Содержимое
clinic/settings.py	Настройки проекта, чтение переменных окружения, параметры безопасности
catalog/models.py	Специальность, врач, услуга, кабинет
schedule/models.py, services.py	График, исключения, функция расчёта свободных интервалов
booking/models.py, services.py	Запись, согласие, уведомление; сервис создания и отмены записи
booking/views.py, templates/booking/	Представления и шаблоны трёх шагов записи
accounts/	Профиль пациента, роли, вход с блокировкой после неудачных попыток
audit/	Модель журнала и функция записи события
*/tests/	Модульные и интеграционные тесты
deploy/	Конфигурация Nginx, служба systemd для Gunicorn, задание cron

Примечание — составлено автором.

Модели данных повторяют логическую схему из подраздела 2.2. Самая важная из них — модель записи, фрагмент которой приведён в листинге 1. Помимо полей в ней объявлено ограничение-исключение: оно создаётся в базе данных миграцией и запрещает пересечение интервалов времени у одного врача для всех записей, кроме отменённых. Конструкция повторяет пример из документации модуля django.contrib.postgres [26].

Листинг 1 – Модель записи на приём (booking/models.py, фрагмент)

```
class TsTzRange(Func):
    function = "TSTZRANGE"
    output_field = DateTimeRangeField()

class Appointment(models.Model):
    class Status(models.TextChoices):
        PLANNED = "planned", "Запланирована"
        DONE = "done", "Состоялась"
        NO_SHOW = "no_show", "Неявка"
        CANCELLED = "cancelled", "Отменена"

    patient = models.ForeignKey(
        "accounts.Patient", on_delete=models.PROTECT)
    doctor = models.ForeignKey(
        "catalog.Doctor", on_delete=models.PROTECT)
    service = models.ForeignKey(
        "catalog.Service", on_delete=models.PROTECT)
    starts_at = models.DateTimeField()
    ends_at = models.DateTimeField()
    status = models.CharField(max_length=12, choices=Status,
                              default=Status.PLANNED)
    comment = models.CharField(max_length=300, blank=True)

    class Meta:
        constraints = [ExclusionConstraint(
            name="no_overlap_per_doctor",
            expressions=[
                (TsTzRange("starts_at", "ends_at", RangeBoundary()),
                 RangeOperators.OVERLAPS),
                ("doctor", RangeOperators.EQUAL),
            ],
            condition=~Q(status="cancelled"),
        )]
```

Ограничение по полю doctor с оператором равенства внутри индекса GiST возможно только при установленном расширении btree_gist. Поэтому первой операцией миграции приложения booking стоит BtreeGistExtension(), которая выполняет CREATE EXTENSION IF NOT EXISTS btree_gist [26, 35]. Для внешних ключей выбрано поведение PROTECT: врача или услугу, у

которых есть записи, нельзя удалить, их можно только сделать неактивными. Иначе удаление врача из справочника стёрло бы историю визитов.

Настройки безопасности собраны в одном месте файла settings.py. Включены USE_TZ и часовой пояс клиники, первым в списке хешеров паролей указан Argon2, cookie сессии и защиты от CSRF передаются только по HTTPS, сервер отдаёт заголовок Strict-Transport-Security. Такой набор соответствует контрольному списку развёртывания, который рекомендует документация Django [27, 29].

3.2 Реализация расчёта расписания и записи на приём

Расчёт свободных интервалов реализован функцией free_slots в модуле schedule/services.py (листинг 2). Функция получает врача, услугу и дату, запрашивает рабочие окна с учётом исключений, одним запросом выбирает все активные записи врача на этот день и затем перебирает возможные начала приёма с шагом 15 минут. Проверка пересечения сделана по классическому условию: два интервала не пересекаются, если один заканчивается не позже, чем начинается другой.

Листинг 2 – Расчёт свободных интервалов (schedule/services.py, фрагмент)

```
STEP = timedelta(minutes=15)
LEAD_TIME = timedelta(hours=1)

def free_slots(doctor, service, day):
    windows = work_windows(doctor, day)   # график + исключения
    busy = list(Appointment.objects
        .filter(doctor=doctor, starts_at__date=day)
        .exclude(status=Appointment.Status.CANCELLED)
        .values_list("starts_at", "ends_at"))
    duration = timedelta(minutes=service.duration_min)
    earliest = timezone.now() + LEAD_TIME
    slots = []
    for w_start, w_end in windows:
        t = w_start
        while t + duration <= w_end:
            end = t + duration
            if t >= earliest and all(end <= b0 or t >= b1
                                     for b0, b1 in busy):
                slots.append(t)
            t += STEP
    return slots
```

Шаг 15 минут выбран как наибольший общий делитель длительностей услуг клиники: первичный приём длится 30 минут, повторный 15, ультразвуковое исследование 45 минут, расширенная консультация 60 минут. При шаге 30 минут каждый повторный приём оставлял бы после себя неиспользуемые пятнадцать минут, а при шаге 5 минут сетка времени на экране становилась бы слишком мелкой. Функция work_windows возвращает моменты времени с часовым поясом, поэтому сравнение с записями из базы выполняется корректно независимо от настроек сервера.

Создание записи реализовано функцией book в модуле booking/services.py (листинг 3). Она работает внутри транзакции transaction.atomic и начинает с блокировки строки врача командой select_for_update. Такая блокировка действует до конца транзакции, и Django требует вызывать её только внутри транзакции, иначе возбуждается исключение TransactionManagementError [28]. Затем повторно проверяется, что интервал входит в график и свободен, создаются запись и событие аудита, а уведомление-подтверждение ставится в очередь.

Листинг 3 – Сервис создания записи (booking/services.py, фрагмент)

```
class SlotTaken(Exception):
    pass

def book(patient, doctor_id, service, starts_at, source="site"):
    ends_at = starts_at + timedelta(minutes=service.duration_min)
    with transaction.atomic():
        doctor = (Doctor.objects.select_for_update()
                  .get(pk=doctor_id))
        if not fits_schedule(doctor, starts_at, ends_at):
            raise SlotTaken("Время вне графика врача")
        if Appointment.objects.filter(
                doctor=doctor,
                starts_at__lt=ends_at, ends_at__gt=starts_at,
        ).exclude(status=Appointment.Status.CANCELLED).exists():
            raise SlotTaken("Это время уже занято")
        try:
            appt = Appointment.objects.create(
                patient=patient, doctor=doctor, service=service,
                starts_at=starts_at, ends_at=ends_at, source=source)
        except IntegrityError:          # сработало ограничение в БД
            raise SlotTaken("Это время уже занято")
        Notification.objects.create(appointment=appt,
                                    kind="confirm")
        audit.write(patient.user, "appointment.create", appt)
    return appt
```

Уведомление не отправляется прямо из функции book. Это сделано намеренно. Если бы письмо уходило внутри транзакции, а сама транзакция потом откатилась, пациент получил бы подтверждение несуществующей записи. Вместо этого в той же транзакции создаётся строка в таблице уведомлений, а отдельная команда send_notifications, которую cron запускает каждую минуту, выбирает неотправленные строки, отправляет письма и отмечает их отправленными. Запись и уведомление о ней либо появляются вместе, либо не появляются вовсе. Та же команда раз в час создаёт напоминания для записей, до которых осталось от 23 до 24 часов.

Обработка исключения IntegrityError нужна на случай, когда проверку в коде обошли, например администратор создал запись напрямую через панель управления в ту же секунду, что и пациент на сайте. Тогда сработает ограничение-исключение в базе данных, транзакция будет отменена, а пользователь получит понятное сообщение вместо страницы ошибки сервера. Представление, вызвавшее book, перехватывает SlotTaken, заново вызывает free_slots и показывает пациенту три ближайших свободных интервала.

3.3 Реализация личного кабинета и рабочих мест сотрудников

Личный кабинет пациента содержит список будущих и прошлых записей. Перенос реализован как отмена старой записи и создание новой в одной транзакции, поэтому если новое время окажется занятым, старая запись сохранится. Отменить или перенести визит можно не позднее чем за два часа до приёма; позже кнопки становятся неактивными, а рядом выводится телефон регистратуры.

Каждое представление, работающее с конкретной записью, ищет её не только по номеру, но и по владельцу (листинг 4). Если пациент подставит в адресную строку номер чужой записи, запрос просто ничего не найдёт и вернёт ответ 404. Это защищает от нарушения контроля доступа, которое возглавляет перечень рисков OWASP [34]. Код ответа 404 выбран вместо 403

сознательно: он не подтверждает постороннему, что запись с таким номером вообще существует.

Листинг 4 – Отмена записи с проверкой владельца (booking/views.py, фрагмент)

```
@login_required
@require_POST
def cancel_appointment(request, pk):
    appt = get_object_or_404(
        Appointment, pk=pk, patient__user=request.user,
        status=Appointment.Status.PLANNED)
    if appt.starts_at - timezone.now() < CANCEL_DEADLINE:
        messages.error(request, "Отменить запись можно не позднее "
                       "чем за 2 часа. Позвоните в регистратуру.")
        return redirect("booking:my")
    cancel(appt, by=request.user)        # сервис: статус + аудит
    messages.success(request, "Запись отменена")
    return redirect("booking:my")
```

Рабочее место администратора построено на отдельной странице «Расписание», доступной группе «Администраторы». Сетка врачей на день формируется одним запросом к базе, а запись по телефону создаётся через ту же функцию book с параметром source="phone". Так все правила (график, длительность, запрет пересечений) действуют одинаково для записей с сайта и из регистратуры. Отметка явки меняет статус записи на «состоялась» или «неявка»; эти данные потом использует отчёт о неявках.

Врач видит только своё расписание. Чужое ему не нужно. В списке пациентов на день выводятся время, фамилия и инициалы, услуга и пометка «впервые», если это первичный приём. Телефон пациента врачу не показывается: для работы на приёме он не нужен, а принцип минимизации требует, чтобы каждый сотрудник видел ровно столько данных, сколько ему нужно [2].

Руководитель клиники работает в стандартной панели администратора Django, в которой настроены формы справочников, фильтры по специальностям и поиск по врачам. Отчёт о загрузке врачей и доле неявок за выбранный период формируется агрегирующим запросом к таблице записей и может быть выгружен в CSV. Сам факт выгрузки фиксируется в журнале аудита, потому что файл содержит сведения об обращениях пациентов.

3.4 Тестирование веб-приложения

Тестирование организовано по уровням так называемой пирамиды тестов: много быстрых модульных тестов в основании, меньше интеграционных тестов с базой данных и небольшое число сквозных сценариев на вершине [38]. Модульные и интеграционные тесты написаны на встроенном в Django тестовом фреймворке, который для каждого теста создаёт отдельную транзакцию и откатывает её после завершения, так что тесты не мешают друг другу [30]. Системные тесты по вариантам использования выполнялись вручную по заранее составленным тестовым случаям.

Тестовые данные подбирались методами разбиения на классы эквивалентности и анализа граничных значений [15]. Для расчёта расписания границы очевидны: приём, который заканчивается ровно в конце смены; приём, который начинается ровно в момент окончания предыдущего; запись менее чем за час до начала. Именно на таких границах чаще всего ошибаются условия «меньше» и «меньше или равно». Пример модульных тестов приведён в листинге 5.

Листинг 5 – Модульные тесты расчёта свободных интервалов (schedule/tests/test_slots.py, фрагмент)

```
class FreeSlotsTests(TestCase):
    def setUp(self):
        self.doctor = make_doctor(start="09:00", end="13:00")
        self.service = make_service(minutes=30)
        self.day = next_weekday(0)     # ближайший понедельник

    def times(self):
        return [localtime(s).time()
                for s in free_slots(self.doctor, self.service,
                                    self.day)]

    def test_last_slot_ends_with_shift(self):
        self.assertEqual(self.times()[-1], time(12, 30))

    def test_neighbours_of_booking(self):
        book(make_patient(), self.doctor.pk, self.service,
             at(self.day, "10:00"))
        slots = self.times()
        self.assertNotIn(time(9, 45), slots)   # пересекается
        self.assertIn(time(9, 30), slots)      # конец в 10:00
        self.assertIn(time(10, 30), slots)     # начало после
```

Отдельный интеграционный тест проверяет одновременную запись. Он унаследован от класса TransactionTestCase, потому что обычный TestCase оборачивает тест в одну транзакцию и не позволяет проверить конкуренцию двух независимых транзакций. В тесте два потока одновременно вызывают book для одного врача и одного времени. Ожидаемый результат — ровно одна созданная запись и одно исключение SlotTaken. Тест прогонялся 200 раз подряд и ни разу не дал двух записей. Для сравнения, при отключённой блокировке select_for_update и удалённом ограничении-исключении двойная запись появлялась примерно в каждом пятом прогоне (результаты прогона на тестовом стенде, условный пример).

Системное тестирование выполнялось на тестовом сервере с копией справочников и вымышленными пациентами. Основные тестовые случаи и результаты их выполнения приведены в таблице 11. Ошибки, найденные при первом прогоне, исправлялись, после чего тестовый случай выполнялся повторно.

Таблица 11 – Тестовые случаи системного тестирования

№	Проверка	Входные данные / действия	Ожидаемый результат	Результат
ТС-01	Запись на свободное время	Терапевт, первичный приём, завтра 10:00	Запись создана, письмо отправлено	Пройден
ТС-02	Запись без согласия	Флажок согласия не отмечен	Сообщение о необходимости согласия, запись не создана	Пройден
ТС-03	Одновременная запись	Два браузера, одно время, нажатие с разницей < 1 с	Одна запись, второму пациенту — соседние интервалы	Пройден
ТС-04	Приём на границе смены	Смена до 13:00, услуга 30 мин	Последний интервал 12:30	Пройден после исправления
ТС-05	Запись в день отпуска врача	Исключение «отпуск» на дату	Дата недоступна в календаре	Пройден
ТС-06	Отмена позже срока	До приёма 1 ч 30 мин	Кнопка неактивна, показан телефон	Пройден
ТС-07	Доступ к чужой записи	Подстановка чужого номера в адрес	Ответ 404	Пройден
ТС-08	Подбор пароля	6 неверных паролей подряд	Блокировка на 15 минут	Пройден
ТС-09	SQL-инъекция в поиске врача	Строка ' OR 1=1 --	Пустой результат, ошибок нет	Пройден
ТС-10	Перевод времени	Сервер в UTC, клиника UTC+3	Время на экране совпадает с графиком	Пройден после исправления
ТС-11	Мобильная вёрстка	Экран 360 × 640	Нет горизонтальной прокрутки	Пройден
ТС-12	Отзыв согласия	Кнопка в личном кабинете	Будущие записи отменены, профиль обезличен	Пройден

Примечание — результаты прогона на тестовом стенде приведены как условный пример.

Две ошибки, найденные при первом прогоне, показательны. Обе — граничные. В тестовом случае ТС-04 последний интервал смены не показывался, потому что в цикле стояло строгое сравнение «меньше» вместо «меньше или равно». В ТС-10 время в письме-напоминании выводилось в UTC: шаблон письма форматировал дату без перевода в местное время. Обе ошибки находились на тех самых границах, на которые были нацелены тестовые случаи.

Найденные дефекты фиксировались в трекере задач репозитория по единому шаблону: шаги воспроизведения, ожидаемый и фактический результат, серьёзность, окружение. Для каждого исправленного дефекта сначала писался автоматический тест, который его воспроизводит, и лишь потом исправлялся код. Благодаря этому набор тестов постепенно превращается в регрессионный: если при следующей доработке кто-то снова поставит строгое сравнение на границе смены, тест укажет на ошибку раньше, чем её заметит пациент. Полный набор из 64 автоматических тестов выполняется примерно за 9 секунд и запускается перед каждым слиянием ветки в main (условные данные для примера).

Нагрузочное тестирование проводилось утилитой Locust по сценарию «открыть каталог — выбрать врача — запросить свободное время — записаться» для 50 одновременных виртуальных пользователей в течение 10 минут. Медианное время ответа страницы выбора времени составило 180 мс, 95-й процентиль — 420 мс, ошибок сервера не было (условные данные для примера). Требование о времени ответа не более 1 секунды выполнено с запасом. Покрытие кода тестами, измеренное утилитой coverage.py, составило 87 % для модулей services.py и 74 % по всему проекту (условные данные для примера).

3.5 Развёртывание и сопровождение

Приложение развёрнуто на виртуальном сервере с операционной системой Linux. Сервер приложения Gunicorn запускается как служба systemd с четырьмя рабочими процессами, что для двух процессорных ядер соответствует распространённой рекомендации «два процесса на ядро». Nginx принимает соединения по HTTPS, отдаёт статические файлы (стили, скрипты, фотографии врачей) напрямую и передаёт остальные запросы Gunicorn через локальный сокет. Сертификат выпускается автоматически и продлевается заданием по расписанию.

Порядок обновления закреплён письменно. Сценарий состоит из пяти шагов: получить новую версию кода из репозитория, установить зависимости в виртуальное окружение, применить миграции базы данных, собрать статические файлы и перезапустить службу Gunicorn. Перед применением миграций сценарий делает внеплановую резервную копию. Если миграция завершается ошибкой, база восстанавливается из этой копии, а служба продолжает работать на старой версии кода. Обновление в рабочее время не проводится: окно для него назначено на вечер воскресенья, когда записей меньше всего.

Таблица 12 – Регламентные задания на сервере

Задание	Периодичность	Назначение
send_notifications	Каждую минуту	Отправка подтверждений, напоминаний и сообщений об отмене из очереди
create_reminders	Каждый час	Постановка в очередь напоминаний за сутки до приёма
pg_dump с шифрованием	Ежедневно в 03:00	Резервная копия базы данных, копирование в объектное хранилище
Удаление старых копий	Ежедневно	Хранение копий не дольше 30 дней
clearsessions	Ежедневно	Удаление истёкших сессий пользователей
Проверка восстановления	Ежемесячно	Развёртывание копии на тестовом сервере и сверка числа записей

Примечание — составлено автором.

Резервное копирование вызывает больше всего вопросов у руководителя. Поэтому оно описано в регламенте отдельно. Копия создаётся утилитой pg_dump, шифруется открытым ключом клиники и только после этого покидает сервер. Закрытый ключ хранится у ответственного за обработку персональных данных, а не на сервере, так что даже при компрометации хранилища копии остаются нечитаемыми. Раз в месяц копия разворачивается на тестовом сервере, и число записей в ней сверяется с рабочей базой: непроверенная резервная копия, по сути, не является копией.

Сопровождение — это прежде всего обновления. Их нельзя откладывать. Django 5.2 получает исправления безопасности до апреля 2028 года [25], и выход каждого такого исправления — повод обновить проект в

ближайшее окно. Остальные пакеты проверяются раз в месяц. Журнал аудита и журналы Nginx хранятся год, после чего удаляются; этот срок закреплён в политике обработки персональных данных клиники, опубликованной на сайте [2].

Для персонала подготовлены две краткие инструкции: для администраторов (запись по звонку, перенос, отметка явки, работа с исключениями графика) и для врачей (просмотр своего дня). Каждая занимает две страницы с изображениями экранов. Обучение администраторов заняло около двух часов, причём большую часть времени ушло не на интерфейс, а на новые правила: например, что запись по телефону для нового пациента невозможно сохранить без отметки о способе получения согласия.

4. Экономическое обоснование проекта

4.1 Расчёт затрат на разработку

Экономическая часть отвечает на простой вопрос руководителя клиники: сколько стоит приложение и через какое время оно окупится. Все суммы в этой главе — условные данные для примера. Они подобраны так, чтобы показать методику расчёта, и перед реальным внедрением должны быть заменены фактическими ставками, тарифами хостинга и показателями клиники.

Трудоёмкость разработки определена методом экспертной оценки по этапам жизненного цикла: для каждого этапа оценено число часов, необходимых одному разработчику. Результат приведён в таблице 13. Наибольшая доля приходится на программирование и отладку, что типично для небольших проектов, где проектирование и реализацию выполняет один человек.

Таблица 13 – Трудоёмкость разработки по этапам

Этап	Трудоёмкость, ч	Доля, %
Анализ предметной области, требования, техническое задание	40	12,5
Проектирование архитектуры, базы данных и интерфейса	48	15,0
Программирование и отладка	160	50,0
Тестирование и исправление ошибок	40	12,5
Развёртывание, документация, обучение персонала	32	10,0
Итого	320	100,0

Примечание — условные данные для примера.

Часовая ставка разработчика рассчитывается делением месячного оклада на месячный фонд рабочего времени:

$$C_ч = \frac{О_{мес}}{Ф_{мес}} = \frac{80\,000}{165} \approx 485\ руб./ч \quad (1)$$

где Сч — часовая ставка, руб./ч; Омес — месячный оклад разработчика, руб. (условно 80 000 руб.); Фмес — месячный фонд рабочего времени, ч (условно 165 ч).

Основная заработная плата за весь проект равна произведению трудоёмкости на часовую ставку:

$$З_{осн} = Т \cdot С_ч = 320 \cdot 485 = 155\,200\ руб. \quad (2)$$

где Т — трудоёмкость разработки, ч.

К основной заработной плате добавляются страховые взносы, накладные расходы и амортизация компьютера, на котором велась разработка. Страховые взносы приняты по единому тарифу 30 % в пределах предельной базы [1]. Накладные расходы (аренда рабочего места, связь, программные инструменты) условно приняты в размере 20 % от основной заработной платы. Амортизация рассчитана для компьютера стоимостью 90 000 руб. со сроком полезного использования три года за два месяца работы над проектом. Полная сумма затрат на разработку определяется по формуле

$$К = З_{осн} + З_{стр} + З_{накл} + А \quad (3)$$

где К — капитальные (единовременные) затраты на разработку, руб.; Зстр — страховые взносы, руб.; Знакл — накладные расходы, руб.; А — амортизация оборудования, руб.

Таблица 14 – Смета затрат на разработку веб-приложения

Статья затрат	Расчёт	Сумма, руб.
Основная заработная плата	320 ч × 485 руб./ч	155 200
Страховые взносы	155 200 × 30 %	46 560
Накладные расходы	155 200 × 20 %	31 040
Амортизация компьютера	90 000 / 36 мес. × 2 мес.	5 000
Итого затраты на разработку		237 800

Примечание — условные данные для примера; ставка страховых взносов — по ст. 425 Налогового кодекса РФ.

Лицензионные платежи в смету не входят. Python, Django, PostgreSQL, Nginx и Bootstrap распространяются по свободным лицензиям, которые разрешают коммерческое использование без отчислений. Это одно из преимуществ выбранного стека перед модулями записи коммерческих медицинских информационных систем, где лицензия обычно оплачивается за каждое рабочее место.

4.2 Затраты на внедрение и эксплуатацию

Внедрение включает аренду и настройку виртуального сервера, перенос справочников врачей и услуг, загрузку графиков работы на первый месяц и обучение персонала. Настройка сервера и обучение уже учтены в этапе «Развёртывание, документация, обучение» таблицы 13, поэтому отдельно они не считаются. Сертификат для HTTPS выпускается бесплатно через центр сертификации Let’s Encrypt, домен у клиники уже есть.

Текущие расходы связаны с хостингом, рассылкой уведомлений и сопровождением. Чтобы выполнить требование о размещении баз данных на территории России, сервер арендуется у российского провайдера. SMS-напоминания отправляются только тем пациентам, которые указали это в профиле, остальным уходят письма. На сопровождение (обновление зависимостей, резервные копии, мелкие доработки) заложено 8 часов в месяц по ставке разработчика с учётом страховых взносов.

Таблица 15 – Годовые эксплуатационные затраты

Статья	**Расчёт**	**Сумма в год, руб.**
Аренда виртуального сервера (2 ядра, 4 ГБ) в российском ЦОД	2 200 руб. × 12 мес.	26 400
Сервис отправки электронной почты	500 руб. × 12 мес.	6 000
SMS-напоминания	2 000 шт. × 3,5 руб. × 12 мес.	84 000
Сопровождение	8 ч × 630 руб./ч × 12 мес.	60 480
Итого		176 880

Примечание — условные данные для примера; 630 руб./ч — ставка 485 руб./ч с учётом страховых взносов 30 %.

Самая крупная статья — SMS, а не сервер. Это может удивить. Если клиника решит отправлять напоминания только по электронной почте или через мессенджер, эксплуатационные расходы снизятся примерно вдвое. Но для пожилых пациентов SMS остаётся самым надёжным каналом, поэтому в базовом расчёте она сохранена.

4.3 Оценка экономического эффекта

Эффект от внедрения складывается из двух составляющих: экономии рабочего времени администраторов и сокращения неявок. Третья возможная составляющая, привлечение новых пациентов за счёт записи вечером и в выходные, в расчёт не включена: её трудно оценить заранее, и расчёт без неё получается осторожнее.

Экономия времени администраторов возникает потому, что часть пациентов будет записываться самостоятельно. Предположим, что из 95 ежедневных обращений по поводу записи около 40 перейдут на сайт, а средняя продолжительность разговора составляет 3 минуты (условные данные для примера). Годовая экономия рассчитывается по формуле

$$Э_{адм} = \frac{N \cdot t \cdot D \cdot C_{адм}}{60} = \frac{40 \cdot 3 \cdot 300 \cdot 355}{60} = 213\,000\ руб. \quad (4)$$

где N — число обращений в день, перешедших на сайт; t — длительность одного разговора, мин; D — число рабочих дней в году; Садм — стоимость часа работы администратора с учётом страховых взносов, руб. (условно 45 000 руб. / 165 ч × 1,3 ≈ 355 руб.).

Освободившиеся два часа в день не означают сокращения ставки. Администратор тратит их на работу с посетителями у стойки, оформление договоров и звонки пациентам, которым нужно перенести приём из-за болезни врача. Для руководителя это время всё равно имеет цену, поэтому в расчёте оно оценено по стоимости часа работы.

Вторая составляющая — сокращение неявок за счёт автоматических напоминаний. Пустое окно в расписании означает, что врач получает зарплату, а клиника не получает выручку. Примем, что в среднем в день планируется 100 визитов, доля неявок снижается с 12 до 9 %, средний чек составляет 2 500 руб., а доля маржинальной прибыли в выручке — 30 % (условные данные для примера). Тогда

$$Э_{неяв} = V \cdot \Delta p \cdot S \cdot D \cdot R \quad (5)$$

где V — плановое число визитов в день; Δp — снижение доли неявок; S — средний чек, руб.; R — доля маржинальной прибыли в выручке. При принятых значениях Энеяв = 100 · 0,03 · 2 500 · 300 · 0,3 = 675 000 руб.

Годовой чистый эффект равен сумме двух составляющих за вычетом эксплуатационных затрат: 213 000 + 675 000 − 176 880 = 711 120 руб. Срок окупаемости единовременных затрат на разработку определяется по формуле

$$T_{ок} = \frac{К}{Э_{адм} + Э_{неяв} - З_{экспл}} = \frac{237\,800}{711\,120} \approx 0{,}33 \ года \quad (6)$$

где Зэкспл — годовые эксплуатационные затраты, руб.

Иначе говоря, при принятых допущениях приложение окупается примерно за четыре месяца. Это быстро. Основные показатели сведены в таблицу 16.

Таблица 16 – Показатели экономической эффективности проекта

Показатель	Значение
Единовременные затраты на разработку, руб.	237 800
Годовые эксплуатационные затраты, руб.	176 880
Экономия времени администраторов, руб./год	213 000
Эффект от сокращения неявок, руб./год	675 000
Чистый годовой эффект, руб.	711 120
Срок окупаемости, мес.	≈ 4
Рентабельность вложений за первый год, %	≈ 199

Примечание — условные данные для примера; рентабельность рассчитана как (711 120 − 237 800) / 237 800.

Наибольшая неопределённость связана с допущением о снижении неявок, ведь именно оно даёт три четверти эффекта. Поэтому выполнен простой анализ чувствительности. Если напоминания снизят долю неявок не на 3, а лишь на 1 процентный пункт, эффект от сокращения неявок составит 225 000 руб., чистый годовой эффект — 261 120 руб., а срок окупаемости вырастет примерно до 11 месяцев. Даже в этом пессимистичном варианте вложения возвращаются в течение первого года, так что проект можно считать экономически оправданным.

Для реальной клиники расчёт стоит повторить через три месяца после запуска, подставив фактические данные из отчёта о неявках, который формирует само приложение. Это позволит руководителю проверить допущения и решить, стоит ли вкладываться в следующие доработки, например в интеграцию с медицинской информационной системой.

ЗАКЛЮЧЕНИЕ

В дипломном проекте разработано веб-приложение для онлайн-записи пациентов в частную медицинскую клинику «[название клиники]». Цель проекта достигнута. Все поставленные задачи решены.

Анализ текущего процесса в нотации IDEF0 показал, что основные потери возникают при поиске свободного времени, двойном вводе данных и ручном обзвоне пациентов накануне визита. Сравнение классов готовых решений показало, что агрегаторы, модули медицинских информационных систем и универсальные конструкторы либо слишком дороги для небольшой клиники без МИС, либо не дают контроля над данными пациентов и правилами записи.

Изучение законодательства привело к нескольким проектным решениям. Сведения о факте обращения составляют врачебную тайну, а данные о здоровье относятся к специальной категории персональных данных, поэтому приложение собирает минимум сведений, разграничивает доступ по ролям, получает от пациента отдельное согласие с сохранением версии его текста и ведёт журнал аудита. Сервер и резервные копии размещаются в России. Для информационной системы определён третий уровень защищённости персональных данных.

Сформулированы двенадцать функциональных требований и нефункциональные требования к производительности, надёжности, безопасности и удобству. Спроектирована трёхзвенная архитектура на Python, Django 5.2 и PostgreSQL 16, логическая модель из одиннадцати сущностей в третьей нормальной форме, алгоритм записи и восемь экранных форм.

Главная техническая задача — исключить двойную запись — решена на двух уровнях. Сервис записи блокирует строку врача на время транзакции и повторно проверяет свободное время, а ограничение-исключение в базе данных не позволяет сохранить пересекающиеся записи даже при обходе проверки в коде. Интеграционный тест с одновременной записью из двух

потоков подтвердил, что при включённой защите двойные записи не возникают.

Тестирование на модульном, интеграционном и системном уровнях выявило две ошибки на граничных значениях, которые были исправлены. Нагрузочный тест показал, что требование ко времени ответа выполняется с запасом. По условным данным экономического расчёта затраты на разработку составляют 237,8 тыс. руб., а срок окупаемости — около четырёх месяцев, в пессимистичном варианте — около года.

Дальнейшее развитие проекта видится в трёх направлениях: интеграция с медицинской информационной системой, если клиника её внедрит; программный интерфейс для внешних агрегаторов; запись по направлениям и комплексным программам обследования, где один визит включает несколько услуг у разных врачей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Налоговый кодекс Российской Федерации (часть вторая) : Федеральный закон от 05.08.2000 № 117-ФЗ. – Ст. 425. Тарифы страховых взносов. – Текст : электронный // Гарант : информационно-правовой портал. – URL: https://base.garant.ru/10900200/f6f70b89a0139a78bc47b22016eb8b75/ (дата обращения: 22.10.2026).

2. Российская Федерация. Законы. О персональных данных : Федеральный закон от 27.07.2006 № 152-ФЗ. – Текст : электронный // RuLaws.ru : [сайт]. – URL: https://rulaws.ru/laws/Federalnyy-zakon-ot-27.07.2006-N-152-FZ/ (дата обращения: 22.10.2026).

3. Российская Федерация. Законы. Об основах охраны здоровья граждан в Российской Федерации : Федеральный закон от 21.11.2011 № 323-ФЗ. – Текст : электронный // КонсультантПлюс : справочная правовая система. – URL: https://www.consultant.ru/law/podborki/vrachebnaya_tajna_jeto/ (дата обращения: 22.10.2026).

4. Об утверждении требований к защите персональных данных при их обработке в информационных системах персональных данных : постановление Правительства Российской Федерации от 01.11.2012 № 1119. – Текст : электронный // ППТ.ру : [сайт]. – URL: https://pravo.ppt.ru/postanovlenie/pravitelstvo/n-1119-2619 (дата обращения: 22.10.2026).

5. Правила предоставления медицинскими организациями платных медицинских услуг : утверждены постановлением Правительства Российской Федерации от 11.05.2023 № 736. – Текст : электронный // Клерк : [сайт]. – URL: https://www.klerk.ru/doc/571044/ (дата обращения: 22.10.2026).

6. Об утверждении Состава и содержания организационных и технических мер по обеспечению безопасности персональных данных при их обработке в информационных системах персональных данных : приказ ФСТЭК России от 18.02.2013 № 21 : зарегистрирован в Минюсте России 14.05.2013 № 28375.

7. Об утверждении федерального государственного образовательного стандарта среднего профессионального образования по специальности 09.02.07 Информационные системы и программирование : приказ Министерства образования и науки Российской Федерации от 09.12.2016 № 1547.

8. ГОСТ 34.602–2020. Информационные технологии. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы : межгосударственный стандарт : введён в действие приказом Росстандарта от 19.11.2021 № 1522-ст : дата введения 2022-01-01.

9. Р 50.1.028–2001. Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования : рекомендации по стандартизации : утверждены Госстандартом России 02.07.2001 : дата введения 2002-07-01. – Москва : ИПК Издательство стандартов, 2001.

10. Буч, Г. Язык UML. Руководство пользователя / Г. Буч, Д. Рамбо, И. Якобсон ; пер. с англ. – 2-е изд. – Москва : ДМК Пресс ; Санкт-Петербург : Питер, 2004. – 432 с. – ISBN 5-94074-260-2.

11. Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт ; перевод с английского и редакция К. А. Птицына. – 8-е изд. – Москва [и др.] : Вильямс, 2005. – 1327 с. – ISBN 5-8459-0788-8.

12. Коннолли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли, К. Бегг ; перевод с английского. – Москва [и др.] : Вильямс, 2003. – 1439 с. – ISBN 5-8459-0527-3.

13. Ларман, К. Применение UML 2.0 и шаблонов проектирования / К. Ларман ; пер. с англ. – 3-е изд. – Москва : Вильямс, 2006. – 736 с.

14. Леоненков, А. В. Самоучитель UML 2 / А. В. Леоненков. – Санкт-Петербург : БХВ-Петербург, 2007. – 576 с. – ISBN 978-5-94157-878-8.

15. Майерс, Г. Искусство тестирования программ / Г. Майерс, Т. Баджетт, К. Сандлер ; перевод с английского и редакция А. Г. Гузикевича. – 3-е изд. – Москва [и др.] : Диалектика, 2012. – 270 с. – ISBN 978-5-8459-1796-6.

16. Моргунов, Е. П. PostgreSQL. Основы языка SQL : учебное пособие / Е. П. Моргунов ; под редакцией Е. В. Рогова, П. В. Лузанова. – Санкт-Петербург : БХВ-Петербург, 2018. – 336 с. – ISBN 978-5-9775-4022-3.

17. Рогов, Е. В. PostgreSQL 16 изнутри / Е. В. Рогов. – Москва : ДМК Пресс, 2024. – 664 с.

18. Фаулер, М. Шаблоны корпоративных приложений : перевод с английского / М. Фаулер. – 2018. – ISBN 978-5-8459-1611-2.

19. Шмуллер, Дж. Освой самостоятельно UML 2 за 24 часа. Практическое руководство / Дж. Шмуллер ; перевод с английского. – Москва : Вильямс, 2005. – 416 с.

20. Chen, P. P.-S. The Entity-Relationship Model – Toward a Unified View of Data / P. P.-S. Chen // ACM Transactions on Database Systems. – 1976. – Vol. 1, № 1. – P. 9–36. – DOI 10.1145/320434.320440.

21. Fielding, R. T. Architectural Styles and the Design of Network-based Software Architectures : dissertation … Doctor of Philosophy in Information and Computer Science / R. T. Fielding. – Irvine : University of California, 2000. – URL: https://ics.uci.edu/~fielding/pubs/dissertation/top.htm (date of access: 22.10.2026).

22. Инфляция вместо роста: что происходит с рынком платных медицинских услуг : [интервью с генеральным директором Eqiva Д. Шубиной]. – Текст : электронный // Коммерсантъ : [сайт]. – 2025. – 24 декабря. – URL: https://www.kommersant.ru/doc/8316041 (дата обращения: 22.10.2026).

23. С 1 сентября оформлять согласие на обработку персональных данных нужно будет по-новому. – Текст : электронный // ППТ.ру : [сайт]. –

URL: https://ppt.ru/news/personal-data/oformlyat-soglasie-na-obrabotku-personalnykh-dannykh-nuzhno-budet-po-novomu (дата обращения: 22.10.2026).

24. Bootstrap 5.3 : documentation. – Текст : электронный // Bootstrap : [сайт]. – URL: https://getbootstrap.com/docs/5.3/layout/grid/ (дата обращения: 22.10.2026).

25. Django 5.2 release notes. – Текст : электронный // Django documentation : [сайт]. – URL: https://docs.djangoproject.com/en/5.2/releases/5.2/ (дата обращения: 22.10.2026).

26. PostgreSQL specific database constraints. – Текст : электронный // Django 5.2 documentation : [сайт]. – URL: https://docs.djangoproject.com/en/5.2/ref/contrib/postgres/constraints/ (дата обращения: 22.10.2026).

27. Password management in Django. – Текст : электронный // Django 5.2 documentation : [сайт]. – URL: https://docs.djangoproject.com/en/5.2/topics/auth/passwords/ (дата обращения: 22.10.2026).

28. QuerySet API reference: select_for_update(). – Текст : электронный // Django 5.2 documentation : [сайт]. – URL: https://docs.djangoproject.com/en/5.2/ref/models/querysets/#select-for-update (дата обращения: 22.10.2026).

29. Security in Django. – Текст : электронный // Django 5.2 documentation : [сайт]. – URL: https://docs.djangoproject.com/en/5.2/topics/security/ (дата обращения: 22.10.2026).

30. Writing and running tests. – Текст : электронный // Django 5.2 documentation : [сайт]. – URL: https://docs.djangoproject.com/en/5.2/topics/testing/overview/ (дата обращения: 22.10.2026).

31. Martin, R. C. The Clean Architecture / R. C. Martin. – Текст : электронный // The Clean Code Blog : [сайт]. – 2012. – 13 August. – URL:

https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html (дата обращения: 22.10.2026).

32. Nielsen, J. 10 Usability Heuristics for User Interface Design / J. Nielsen. – Текст : электронный // Nielsen Norman Group : [сайт]. – URL: https://www.nngroup.com/articles/ten-usability-heuristics/ (дата обращения: 22.10.2026).

33. Password Storage Cheat Sheet. – Текст : электронный // OWASP Cheat Sheet Series : [сайт]. – URL: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html (дата обращения: 22.10.2026).

34. OWASP Top 10:2021. – Текст : электронный // OWASP Foundation : [сайт]. – URL: https://top10.owasp.org/2021/ (дата обращения: 22.10.2026).

35. PostgreSQL 16 Documentation. 5.4. Constraints. 5.4.6. Exclusion Constraints. – Текст : электронный // PostgreSQL : [сайт]. – URL: https://www.postgresql.org/docs/16/ddl-constraints.html (дата обращения: 22.10.2026).

36. PostgreSQL 16 Documentation. 11.8. Partial Indexes. – Текст : электронный // PostgreSQL : [сайт]. – URL: https://www.postgresql.org/docs/16/indexes-partial.html (дата обращения: 22.10.2026).

37. Fielding, R. RFC 9110. HTTP Semantics / R. Fielding, M. Nottingham, J. Reschke. – Текст : электронный // RFC Editor : [сайт]. – 2022. – URL: https://www.rfc-editor.org/rfc/rfc9110 (дата обращения: 22.10.2026).

38. Vocke, H. The Practical Test Pyramid / H. Vocke. – Текст : электронный // martinfowler.com : [сайт]. – 2018. – URL: https://martinfowler.com/articles/practical-test-pyramid.html (дата обращения: 22.10.2026).

39. Wiggins, A. The Twelve-Factor App / A. Wiggins. – Текст : электронный. – URL: https://12factor.net/ru/config (дата обращения: 22.10.2026).

40. Web Content Accessibility Guidelines (WCAG) 2.1 : W3C Recommendation. – Текст : электронный // W3C : [сайт]. – URL: https://www.w3.org/TR/WCAG21/ (дата обращения: 22.10.2026).

ПРИЛОЖЕНИЕ А. СКРИПТ СОЗДАНИЯ ОСНОВНЫХ ТАБЛИЦ БАЗЫ ДАННЫХ

Скрипт сформирован командой sqlmigrate по миграциям Django и сокращён: служебные таблицы Django и часть индексов опущены.

```
CREATE EXTENSION IF NOT EXISTS btree_gist;

CREATE TABLE specialty (
    id          bigserial PRIMARY KEY,
    name        varchar(100) NOT NULL UNIQUE,
    description text NOT NULL DEFAULT ''
);

CREATE TABLE doctor (
    id               bigserial PRIMARY KEY,
    user_id          integer UNIQUE REFERENCES auth_user (id),
    specialty_id     bigint NOT NULL REFERENCES specialty (id),
    last_name        varchar(60) NOT NULL,
    first_name       varchar(60) NOT NULL,
    middle_name      varchar(60) NOT NULL DEFAULT '',
    education        text NOT NULL,
    qualification    text NOT NULL,
    experience_years smallint CHECK (experience_years >= 0),
    is_active        boolean NOT NULL DEFAULT true
);

CREATE TABLE service (
    id           bigserial PRIMARY KEY,
    name         varchar(150) NOT NULL,
    duration_min smallint NOT NULL
                 CHECK (duration_min > 0 AND duration_min % 15 = 0),
    price        numeric(10, 2) NOT NULL CHECK (price >= 0),
    is_primary   boolean NOT NULL DEFAULT false
);

CREATE TABLE doctor_service (
    doctor_id  bigint REFERENCES doctor (id),
    service_id bigint REFERENCES service (id),
    PRIMARY KEY (doctor_id, service_id)
);

CREATE TABLE cabinet (
    id     bigserial PRIMARY KEY,
    number varchar(10) NOT NULL UNIQUE,
    floor  smallint NOT NULL
);

CREATE TABLE work_schedule (
    id         bigserial PRIMARY KEY,
    doctor_id  bigint NOT NULL REFERENCES doctor (id),
    cabinet_id bigint NOT NULL REFERENCES cabinet (id),
    weekday    smallint NOT NULL CHECK (weekday BETWEEN 0 AND 6),
    starts     time NOT NULL,
    ends       time NOT NULL CHECK (ends > starts)
```

```
);

CREATE TABLE schedule_exception (
    id        bigserial PRIMARY KEY,
    doctor_id bigint NOT NULL REFERENCES doctor (id),
    day       date NOT NULL,
    kind      varchar(10) NOT NULL
              CHECK (kind IN ('vacation', 'sick', 'extra')),
    starts    time,
    ends      time
);

CREATE TABLE patient (
    id             bigserial PRIMARY KEY,
    user_id        integer UNIQUE REFERENCES auth_user (id),
    last_name      varchar(60) NOT NULL,
    first_name     varchar(60) NOT NULL,
    middle_name    varchar(60) NOT NULL DEFAULT '',
    phone          varchar(16) NOT NULL,
    email          varchar(254),
    birth_date     date,
    sms_allowed    boolean NOT NULL DEFAULT false,
    is_anonymized boolean NOT NULL DEFAULT false
);
CREATE TABLE appointment (
    id         bigserial PRIMARY KEY,
    patient_id bigint NOT NULL REFERENCES patient (id),
    doctor_id  bigint NOT NULL REFERENCES doctor (id),
    service_id bigint NOT NULL REFERENCES service (id),
    starts_at  timestamptz NOT NULL,
    ends_at    timestamptz NOT NULL CHECK (ends_at > starts_at),
    status     varchar(12) NOT NULL DEFAULT 'planned'
               CHECK (status IN ('planned', 'done',
                                 'no_show', 'cancelled')),
    source     varchar(8) NOT NULL DEFAULT 'site'
               CHECK (source IN ('site', 'phone')),
    comment    varchar(300) NOT NULL DEFAULT '',
    created_at timestamptz NOT NULL DEFAULT now(),
    CONSTRAINT no_overlap_per_doctor EXCLUDE USING gist (
        doctor_id WITH =,
        tstzrange(starts_at, ends_at) WITH &&
    ) WHERE (status <> 'cancelled')
);

CREATE UNIQUE INDEX appointment_doctor_start_active
    ON appointment (doctor_id, starts_at)
    WHERE status <> 'cancelled';

CREATE TABLE consent_text (
    id         bigserial PRIMARY KEY,
    version    varchar(10) NOT NULL UNIQUE,
    body       text NOT NULL,
    valid_from date NOT NULL
);

CREATE TABLE consent (
    id              bigserial PRIMARY KEY,
    patient_id      bigint NOT NULL REFERENCES patient (id),
```

```
    consent_text_id bigint NOT NULL REFERENCES consent_text (id),
    given_at        timestamptz NOT NULL DEFAULT now(),
    ip_address      inet,
    revoked_at      timestamptz
);

CREATE TABLE notification (
    id             bigserial PRIMARY KEY,
    appointment_id bigint NOT NULL REFERENCES appointment (id),
    kind           varchar(10) NOT NULL
                   CHECK (kind IN ('confirm', 'remind', 'cancel')),
    channel        varchar(5) NOT NULL DEFAULT 'email',
    created_at     timestamptz NOT NULL DEFAULT now(),
    sent_at        timestamptz
);

CREATE TABLE audit_log (
    id          bigserial PRIMARY KEY,
    user_id     integer REFERENCES auth_user (id),
    action      varchar(40) NOT NULL,
    object_type varchar(40) NOT NULL,
    object_id   bigint,
    ip_address  inet,
    created_at  timestamptz NOT NULL DEFAULT now()
);
```

ПРИЛОЖЕНИЕ Б. ВЫДЕРЖКИ ИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ

Таблица 17 – Основные положения технического задания на веб-приложение

Раздел	Содержание
Наименование системы	Веб-приложение для онлайн-записи пациентов клиники «[название клиники]»
Назначение	Самостоятельная запись пациентов на приём через сайт, ведение расписания врачей и учёт явки
Цели создания	Сократить число звонков по поводу записи; исключить двойную запись; снизить долю неявок; упорядочить доступ к данным пациентов
Пользователи	Пациенты; администраторы регистратуры (2); врачи (9); руководитель клиники (1)
Функции	ФТ-1 – ФТ-12 (таблица 4 пояснительной записки)
Требования к производительности	Время ответа страницы выбора времени не более 1 с при 50 одновременных пользователях
Требования к надёжности	Ежедневное резервное копирование, хранение копий 30 дней, восстановление не более 4 ч
Требования к защите информации	Третий уровень защищённости персональных данных; HTTPS; ролевой доступ; журнал аудита; размещение БД в РФ
Требования к эргономике	Запись не более чем за 3 шага; адаптивная вёрстка от 360 пикселей; контраст текста не ниже 4,5 : 1
Программное обеспечение	Python 3.12, Django 5.2, PostgreSQL 16, Nginx, Gunicorn, ОС Linux
Технические средства	Виртуальный сервер: 2 ядра, 4 ГБ оперативной памяти, 40 ГБ SSD
Порядок приёмки	Системное тестирование по тестовым случаям ТС-01 – ТС-12, опытная эксплуатация 2 недели

Примечание — составлено автором; численность персонала — условные данные для примера.