**СОДЕРЖАНИЕ**

**ВВЕДЕНИЕ** 3

1. Теоретический раздел 6

1.1 Основы автоматизации процессов в поликлиниках: задачи, преимущества и особенности 6

1.2 Принципы организации электронной очереди и онлайн-записи пациентов 8

1.3 Роль уведомлений и напоминаний в повышении удобства пациентов 10

1.4 Обеспечение конфиденциальности и защиты данных пациентов в информационных системах 11

1.5 Понятие и значение системы электронной очереди 12

1.6 Преимущества и недостатки электронных очередей 20

1.7 Теоретические основы проектирования программного обеспечения 21

2. Исследовательский раздел 24

2.1 Анализ существующих систем электронной очереди и онлайн-записи для медицинских учреждений 24

2.2 Исследование потребностей поликлиниках и пациентов в автоматизации записи и отслеживания очередей 28

2.3. Формирование функциональных требований к системе 31

2.4. Обоснование выбора технологий 32

3. Глава 3. Практический раздел 36

3.1. Проектирование структуры базы данных для хранения информации о пациентах, записях и очередях 36

3.2. Разработка серверной части системы для онлайн-записи, управления очередью и отправки уведомлений 37

3.3. Создание пользовательского интерфейса для онлайн-записи, просмотра текущей очереди и статусов 39

3.4. Внедрение функционала для отслеживания очереди в реальном времени 41

3.5. Настройка модуля SMS-уведомлений для информирования пациентов о статусе записи, времени приема и напоминаниях 42

3.6. Проведение тестирования системы: проверка корректности записи, управления очередью и отправки уведомлений 43

Заключение 47

Список использованных источников 48

Приложения 50

**ВВЕДЕНИЕ**

Разработка системы электронной очереди для поликлиники, включающей онлайн-запись, отслеживание очереди и SMS-уведомления для пациентов, представляет собой интегрированное решение для управления процессами записи на прием в медицинских учреждениях. Оно предназначено для повышения эффективности работы поликлиник и улучшения качества обслуживания пациентов. Оно будет представлять собой мощный инструмент для оптимизации работы поликлиник, а также создания комфортных условий для пациентов, облегчая их взаимодействие с медицинскими учреждениями.

Актуальность темы исследования. В последние годы наблюдается устойчивый рост числа обращений граждан в медицинские учреждения, что приводит к увеличению нагрузки на поликлинику и затруднениям в управлении потоками пациентов. В условиях ограниченных ресурсов и необходимости эффективного использования времени медицинских работников, разработка системы не только целесообразной, но и необходимой. Во-первых, внедрение онлайн-записи позволяет пациентам записываться на прием в удобное для них время, что значительно снижает количество посетителей, ожидающих свою очередь в поликлиниках. Это, в свою очередь, ведет к уменьшению времени ожидания и создает более комфортные условия для обращения за медицинской помощью. Во-вторых, система отслеживания очереди предоставляет пациентам возможность видеть свою позицию очереди и ориентироваться во времени ожидания, что способствует снижению стресса и неопределенности.

Важным аспектом является SMS-уведомления. Они информируют пациентов о времени их записи, возможных изменениях в расписании или необходимости подготовки к приему. Кроме того, такие системы способствуют улучшению общей организационной структуры работы поликлиник, позволяя медицинскому персоналу планировать свою работу более эффективно и уделять внимание тем вопросам, которые требуют большего взаимодействия с пациентами. Этим обусловлены актуальность выбора данной темы.

В итоге, разработка системы электронной очереди для поликлиники является актуальной темой исследования, способствующей модернизации процессов управления пациентами, повышению доступности и качества медицинских услуг.

Актуальность темы курсовой работы определила цель и задачи исследования, объект и предмет.

Целью курсовой работы является создание интегрированного программного решения, которое существенно повысит эффективность работы поликлиники и улучшит качество обслуживания пациентов.

Реализация данной цели исследования обусловила необходимость решения следующих задач:

1. Анализ существующих решений и методов управления очередями в медицинских учреждениях;
2. Исследование требований пользователей к удобству использования и доступности системы;
3. Разработка архитектуры, обеспечивающей высокую степень надежности и безопасности данных;
4. Реализация программного обеспечения, включающего функции онлайн-записи, оповещение пациентов и мониторинга очереди;
5. Тестирование системы с реальными пользователями для получения обратной связи и оценки её эффективности.

Объектом исследования является процесс управления очередями в медицинских учреждениях, с акцентом на взаимодействие между пациентами и медиками.

Предметом исследования – автоматизация управления очередями и записи на прием в медицинских учреждениях.

Методология исследования представляет собой структурированный подход, включающий последовательность этапов, методов и инструментов, направленных на достижение поставленных целей.

Степень разработанности проблемы можно охарактеризовать как находящуюся на высоком уровне, однако с учетом постоянных изменений и нововведений в сфере здравоохранения, данная проблема требует глубокого изучения.

Теоретические исследования по проблемам, можно сказать, что внедрение систем электронной очереди часто сталкивается с рядом проблем, включая недостаточную техническую базу, сопротивление со стороны персонала и пациентов, а также ряд организационных вопросов.

Информационной базой исследования послужили монографии и статьи по медицинской информатике, которые анализируют современные технологии управления очередями в здравоохранении и их влияние на качество обслуживания пациентов. Еще служат интернет-ресурсы. Эта информационная база является основополагающей для углубленного понимания проблемы, ее контекста и разработки эффективного решения в рамках проекта.

Практическая значимость исследования заключается в выводах и предложениях по результатам исследования поставленной в курсовой работе проблемы, сделанных на основе исследования учебных пособий, монографий и статей современных авторов, статистических и аналитических материалов.

**Глава 1. Теоретический раздел**

В данном разделе будут рассмотрены основные принципы и подходы к системе электронной очереди с SMS-уведомлениями. Для улучшения жизнедеятельности, как пациентам, так и врачам. Особое внимание уделяется новым технологиям, которые позволяют быстрее автоматизировать процессы электронной очереди и повысить эффективность использования на современном уровне человечества.

**1.1 Основы автоматизации процессов в поликлиниках: задачи, преимущества и особенности**

Автоматизация процессов в поликлиниках – это внедрение информационных технологий и программных систем для упрощения оптимизации работы медицинских учреждений. Это позволяет повысить качество обслуживания пациентов, снизить административную нагрузку на медицинский персонал и улучшить организацию рабочих процессов. В дальнейшем для упрощения всей жизни и переход на интернет-ресурсы.

Ниже приведены задачи с примерами, для понимания работы курсовой:

**Задачи:**

1. Управление записями на прием

* Внедрение систем онлайн-записи для удобства
* Автоматизация подтверждения записей и уведомлений о предстоящих приемах

1. Отслеживание очередей

* Внедрение электронных медицинских карт для хранения и обработки информации о пациентах
* Информирование пациентов о статусе их записи и ориентировочном времени приема

1. Управление медицинскими данными

* Внедрение электронных медицинских карт для хранения и обработки информации о пациентах
* Упрощение доступа к медицинским данным и истории болезней

1. Анализ и отчетность

* Сбор и анализ статистических данных о посещаемости, пропусках приемов и производительности медицинского персонала
* Генерация отчетов для руководства и анализа эффективности работы поликлиники

**Преимущества:**

1. Увеличение эффективности (уменьшение времени, затрачиваемого на административные задачи)
2. Улучшение качества обслуживания (пациенты получают возможность записываться на прием в удобное для них время)
3. Снижение вероятности ошибок (автоматизация процессов снижает риск ошибок, связанных с ручным вводом данных или неверным планированием)
4. Быстрый доступ к информации (медицинский персонал получает мгновенный доступ к информации о пациентах, что способствует более качественному лечению)
5. Повышение прозрачности и ответственности (более простая отчетность и анализ процессов позволяют выявить проблемы и повышать уровень ответственности сотрудников)
6. Экономия времени и ресурсов (сокращение времени ожидания для пациентов и снижение нагрузки на инфраструктуру мед. учреждений)

**Особенности:**

1. Автоматизация должна обеспечивать совместимость и интеграцию с существующими информационными системами в поликлиниках;
2. Специальное внимание должно уделяться защите персональных данных;
3. Обеспечивать техническую поддержку.

Таким образом, автоматизация процессов в поликлиниках представляет важный шаг к улучшению качества медицинских услуг, повышению эффективности работы учреждений и созданию более комфортных условий для пациентов.

**1.2. Принципы организации электронной очереди и онлайн-записи пациентов**

Организация электронной очереди и онлайн-записи в медицинских учреждениях – важный аспект современного здравоохранения, который позволяет оптимизировать процессы обслуживания и повысить уровень удовлетворенности пациентов.

**Принципы организации:**

1. Удобность и доступность:

* Система должна иметь интуитивно приятный и простой в использовании интерфейс, позволяющий пользователям легко ориентироваться и записываться на прием;
* Пациенты должны иметь возможность записываться на прием онлайн.

1. Прозрачность процессов:

* Пациенты должны видеть текущую очередь и ориентировочное время ожидания;
* Возможность оставить отзыв о качестве обслуживания после посещения.

1. Безопасность данных:

* Система должна соответствовать актуальным нормам законодательства о защите данных и гарантировать конфиденциальность информации
* Необходимо ограничить доступ к данным для неавторизованных пользователей и обеспечить доступ только тем сотрудникам, кому это необходимо

1. Интеграция с существующими системами:

* Система электронной очереди должна быть совместима с электронной медицинской картой для обмена данными и доступа к информации

1. Гибкость и адаптивность:

* Система должна обеспечивать возможность настройки под конкретные нужды поликлиники и ее пациентов
* Возможность добавлять новые функции и улучшений в систему на основе обратной связи

1. Анализ и оптимизация

* Система должна вести учет обращений и оценивать эффективность работы
* Регулярный анализ полученной информации и отзывов пользователей

**1.3. Роль уведомлений и напоминаний в повышении удобства пациентов**

Уведомления и напоминания играют ключевую роль в повышении удобства пациентов при использовании систем электронной очереди и онлайн-записи в поликлиниках.

Основные их значимости:

* Снижение количества пропусков приемов: напоминания помогают пациентам не забыть о визите к врачу. Например, система смс-напоминаний может отправлять сообщения за 24 часа до приема, а также за несколько часов до него, как это делает программа Dentist Plus для стоматологических клиник.
* Повышение уровня информированности: уведомления могут содержать важную информацию о необходимых подготовительных действиях перед приемом. Например, о необходимости голодания или прекращения приема определенных препаратов.
* Удобство в изменении записей: если необходимо изменить время или дату приема, напоминания позволяют сделать это оперативно без дополнительных хлопот для пациента — например, через ответное смс-сообщение или WhatsApp.
* Уменьшение стресса и тревожности: напоминающие сообщения могут успокоить тех пациентов, кто испытывает тревогу перед посещением врача. Зная точно, дату и время приема заранее помогает снизить уровень беспокойства.
* Возможность обратной связи: многие системы позволяют получать ответы от клиентов на смс-уведомления. Например, подтверждение или перенос записи, что упрощает взаимодействие между врачами и их клиентами.
* Экономия времени персонала медучреждений: системы автоматических напоминаний снижают нагрузку на административный персонал больниц по сравнению с ручными звонками для подтверждения записей.
* Персонализация обслуживания: уведомления могут быть адаптированы под конкретные потребности каждого пациента — например включать рекомендации по здоровью на основе истории болезни человека.

**1.4. Обеспечение конфиденциальности и защиты данных пациентов в информационных системах**

Обеспечение конфиденциальности и защиты пациентов в информационных системах здравоохранения – это сложный и многоаспектный процесс, который требует комплексного подхода.

Он включает в себя юридические, технические и организационные меры, направленные на защиту личных и медицинских данных пациентов от несанкционированного доступа, утечки или повреждения.

**Основные:**

1. Законодательные и нормативные требования

* Соблюдение законов о защите данных
* Уведомление о нарушениях
* Конституция Российской Федерации: статьи 22-24 защищают право на неприкосновенность частной жизни, личной и семейной тайны.
* Федеральный закон от 27.07.2006 № 152-ФЗ «О персональных данных»: регулирует отношения по обработке персональных данных, включая сбор, хранение и уничтожение.
* Федеральный закон от 21.11.2011 № 323-ФЗ «Об основах охраны здоровья граждан в Российской Федерации»: устанавливает правила обращения со сведениями о состоянии здоровья пациента.

Подходящие меры могут существенно снизить утечек данных и создать доверительную атмосферу между пациентами и медицинскими учреждениями, что критически важно для обеспечения безопасности и качества предоставляемых услуг.

**1.5. Понятие и значение системы электронной очереди**

Понятие системы электронной очереди.  
Система электронной очереди представляет собой комплексное цифровое решение, которое коренным образом преобразует традиционный процесс организации приема посетителей в медицинских учреждениях[[1]](#footnote-1). На технологическом уровне это совокупность взаимосвязанных программных модулей и аппаратных компонентов, образующих единую экосистему управления потоками пациентов.

Инициализация БД:

DatabaseHelper.cs → метод InitializeDatabase()

Процесс записи:

DoctorsForm.cs → обработчик клика по кнопке "Запись"

Структура данных:

SQL-запросы создания таблиц в DatabaseHelper.InitializeDatabase()

В медицинском контексте такие системы выполняют роль цифрового администратора, который берет на себя функции координации между тремя ключевыми участниками процесса: пациентом, медицинским персоналом и административными работниками. Технологическая основа системы строится на специализированном программном обеспечении, которое может быть реализовано как облачное решение или как локально устанавливаемый продукт.

Архитектурно система включает несколько обязательных элементов: серверную часть для обработки данных, клиентские интерфейсы для разных категорий пользователей, базу данных для хранения информации и модули интеграции с внешними сервисами. Современные реализации часто используют микросервисную архитектуру, что обеспечивает гибкость и масштабируемость решения.

Важной характеристикой таких систем является их адаптивность - они могут настраиваться под специфику конкретного медицинского учреждения, учитывая количество специалистов, график их работы, особенности потоков пациентов и другие параметры. Некоторые продвинутые системы включают элементы искусственного интеллекта для прогнозной аналитики и оптимизации расписания.

**Система может включать несколько компонентов:**

Интерфейс для записи на прием - не просто форма выбора времени, а многофункциональный инструмент, который может предлагать интеллектуальные рекомендации по выбору специалиста, показывать рейтинги врачей, предоставлять информацию о подготовке к приему и даже интегрироваться с системами онлайн-оплаты услуг.

Мониторинг очереди - сложный механизм реального времени, который учитывает не только порядок записи, но и текущую загруженность врачей, возможные задержки и форс-мажорные обстоятельства, динамически пересчитывая время приема.

Уведомления - многоуровневая система оповещений, которая может включать SMS-сообщения, email-рассылки, push-уведомления в мобильных приложениях, а также интеграцию с популярными мессенджерами.

Отчет и аналитика - мощный инструмент для администрации, предоставляющий не только статистические данные, но и прогнозные модели, выявляющий сезонные закономерности и помогающий в планировании ресурсов.

**Значение системы электронной очереди:**

* Оптимизация времени ожидания - система использует сложные алгоритмы распределения пациентов, учитывающие множество факторов.

Пример: в коде DoctorsForm реализована таблица с расписанием, где пациент видит только свободные слоты (кнопки "Запись"). Алгоритм в GetAvailableSlots() исключает уже занятые времена через LEFT JOIN:

**sql**

SELECT s.time FROM schedule s

LEFT JOIN appointments a ON s.schedule\_id = a.schedule\_id

WHERE s.doctor\_id = 3 AND a.schedule\_id IS NULL

* Улучшение качества обслуживания - помимо базового удобства, система способствует созданию предсказуемой среды для пациентов и стабильных рабочих условий для врачей, что в совокупности повышает удовлетворенность от медицинского обслуживания.

Пример: при регистрации (AuthForm) проверяется возраст пациента:

csharp

if (DateTime.Today.Year - birthDatePicker.Value.Year < 14) {

MessageBox.Show("Пациенты младше 14 лет не могут записываться самостоятельно");

}

* Эффективное управление ресурсами - администрация получает инструменты для балансировки нагрузки между специалистами, возможности оперативного перераспределения ресурсов и выявления "узких мест" в работе учреждения.

Пример: администратор может добавлять новые слоты через прямой SQL-запрос:

**sql**

INSERT INTO schedule (doctor\_id, time)

VALUES (2, '15:00') ON CONFLICT (doctor\_id, time) DO NOTHING

* Снижение административной нагрузки - автоматизация рутинных операций высвобождает до 30-40% рабочего времени административного персонала, позволяя перераспределить эти ресурсы на более важные задачи.

Пример: автоматическое создание записи через CreateAppointment() без участия администратора:

csharp

dbHelper.CreateAppointment(userId, scheduleId);

* Сбор и анализ данных - накапливаемая статистика становится ценным активом для клиники, позволяя выявлять сезонные колебания спроса, оценивать эффективность работы специалистов и оптимизировать бизнес-процессы.

Пример: для статистики можно выполнить запрос:

sql

SELECT d.name, COUNT(a.appointment\_id)

FROM doctors d

LEFT JOIN schedule s ON d.doctor\_id = s.doctor\_id

LEFT JOIN appointments a ON s.schedule\_id = a.schedule\_id

GROUP BY d.name

* Прозрачность и доверие - объективная система распределения времени приема исключает субъективный фактор и создает равные условия для всех пациентов, повышая доверие к медицинскому учреждению.

Пример: в ConfirmAppointmentForm пациент видит точное время и врача перед подтверждением:

csharp

Label.Text = $"Врач: {doctorId}, Время: {time}";

* Упрощение процесса записи - современные интерфейсы делают запись интуитивно понятной даже для технически неподготовленных пользователей, а мультиканальность доступа обеспечивает удобство для разных категорий пациентов.

Пример: в DoctorsForm реализована пошаговая запись:

Выбор врача

Выбор свободного слота

Подтверждение

* Снижение затрат - оптимизация человеческих ресурсов, сокращение простоев и повышение общей эффективности работы учреждения приводят к значительной экономии операционных расходов.

Пример: отсутствие необходимости в бумажных журналах благодаря таблице appointments.

* Устойчивость к изменениям - модульная архитектура позволяет оперативно адаптировать систему к новым требованиям, будь то изменения в законодательстве, эпидемиологические меры или внутренние регламенты учреждения.

Пример: добавление нового поля в таблицу:

sql

ALTER TABLE doctors ADD COLUMN room\_number INTEGER;

* Контроль соблюдения графика - интеллектуальные механизмы отслеживают соблюдение временных интервалов как со стороны пациентов, так и со стороны медицинского персонала, минимизируя накладки и задержки.

Пример: триггер автоматически обновляет статус слота:

sql

UPDATE schedule SET is\_available = FALSE

WHERE schedule\_id = NEW.schedule\_id;

* Поддержка дистанционного обслуживания - интеграция с телемедицинскими платформами создает единое пространство для оказания медицинских услуг, сочетая очные и дистанционные форматы взаимодействия.

Пример: интеграция могла бы выглядеть как добавление поля:

sql

ALTER TABLE appointments ADD COLUMN is\_remote BOOLEAN DEFAULT FALSE;

* Автоматизация напоминаний - персонализированная система уведомлений значительно сокращает количество пропущенных приемов, повышая эффективность использования медицинских ресурсов.

Пример: потенциальный запрос для выборки предстоящих записей:

sql

SELECT u.phone, d.name, s.time

FROM appointments a

JOIN users u ON a.user\_id = u.user\_id

JOIN schedule s ON a.schedule\_id = s.schedule\_id

JOIN doctors d ON s.doctor\_id = d.doctor\_id

WHERE s.time BETWEEN NOW() AND NOW() + INTERVAL '1 day'

* Мультиплатформенность - единая система доступна через различные каналы взаимодействия, обеспечивая непрерывность сервиса и удобство для пользователей с разными цифровыми привычками.

Пример: в текущей системе это реализовано через единую БД, доступную с разных клиентов.

* Повышение репутации учреждения - внедрение современной системы электронной очереди позиционирует клинику как технологически продвинутую организацию, что становится конкурентным преимуществом на рынке медицинских услуг.

Пример: профессиональный интерфейс AppointmentSuccessForm.cs с подтверждением:

csharp

Label.Text = "Запись подтверждена!";

* Снижение стресса для пациентов - четкое понимание времени приема и минимизация времени ожидания в коридорах создает более комфортную психологическую атмосферу для посетителей.

Пример: точное время в ConfirmAppointmentForm.cs исключает неопределенность:

csharp

Label.Text = $"Вы записаны на {time}";

* Улучшение условий труда персонала - предсказуемый график и равномерная нагрузка способствуют снижению профессионального выгорания среди медицинских работников.

Пример: равномерное распределение нагрузки через:

sql

SELECT doctor\_id, COUNT(\*)

FROM schedule

WHERE is\_available = FALSE

GROUP BY doctor\_id;

* Экологичность - цифровизация процессов сокращает использование бумажных носителей, внося вклад в экологическую политику учреждения.

Пример: полный отказ от бумажных носителей в пользу таблицы appointments.

* Возможности для развития - накопленные данные и опыт использования системы создают основу для дальнейшей цифровой трансформации медицинского учреждения.

Пример: накопленные данные в appointments позволяют анализировать:

sql

SELECT EXTRACT (HOUR FROM s.time) as hour, COUNT(\*)

FROM appointments a

JOIN schedule s ON a.schedule\_id = s.schedule\_id

GROUP BY hour

ORDER BY hour;

Система электронной очереди перестает быть просто инструментом организации очереди, становясь ключевым элементом цифровой инфраструктуры современной клиники. Она создает технологическую основу для трансформации всего цикла взаимодействия с пациентом - от первичной записи до завершения лечения и последующего наблюдения. В условиях растущих требований к качеству медицинских услуг и эффективности работы учреждений, такие системы становятся не просто полезным дополнением, а необходимым условием успешной работы.

Перспективы развития таких решений связаны с углубленной интеграцией с электронными медицинскими картами, системами поддержки врачебных решений, инструментами предиктивной аналитики и другими цифровыми компонентами "умной" клиники. Это создает предпосылки для принципиально нового уровня организации медицинской помощи, где электронная очередь становится лишь видимой частью сложной, но хорошо отлаженной системы управления здоровьем пациентов.

**1.6. Преимущества и недостатки электронных очередей**

Системы электронной очереди (ЭО) представляют собой важный инструмент для управления потоками клиентов в различных сферах, включая здравоохранение, банки, государственные учреждения и торговые организации. Существует множество преимуществ и недостатков таких систем, которые стоит учитывать при их внедрении. На основании исследовании пункта о системе электронной очереди, были выявлены: преимущества и недостатки

**Преимущества:**

* Сниженное время ожидания: электронные очереди (ЭО) позволяют пациентам записываться на прием заранее, что значительно сокращает время ожидания;
* Удобство записи: запись через веб-сайты или мобильные приложения доступна в любое время, что повышает гибкость и доступность сервиса;
* Улучшение качества обслуживания: ЭО способствует более организованному обслуживанию, что увеличивает удовлетворенность клиентов;
* Оптимизация ресурсов: системы позволяют эффективно распределять рабочее время сотрудников, избегая перегрузок;
* Сбор данных для анализа: ЭО обеспечивает сбор статистических данных для анализа работы и оптимизации процессов.

**Недостатки:**

* Технические сбои: ЭО подвержены сбоям, что может привести к неудобствам для клиентов;
* Капитальные затраты на внедрение: внедрение системы требует первоначальных инвестиций и обучения персонала;
* Проблемы с защитой данных: системы должны соответствовать требованиям по защите данных; утечки могут негативно сказаться на репутации;
* Недостаток человеческого взаимодействия: автоматизация может снизить личный контакт между клиентами и персоналом, что не всегда воспринимается положительно.

**1.7. Теоретические основы проектирования программного обеспечения**

Проектирование программного обеспечения — это комплексный процесс, направленный на создание эффективных, надежных и удобных в использовании программных систем. Основные теоретические основы включают несколько ключевых аспектов и принципов.

1. **Требования к системе**

* Сбор требований: это этап сбора информации о функциональных и нефункциональных требованиях системы для понимания ожиданий пользователей и существующих ограничений.
* Анализ требований: проверка собранных требований на согласованность и осуществимость с использованием техник, таких как интервью с пользователями, прототипирование или мозговой штурм.

1. **Архитектурное проектирование**

* Архитектурные шаблоны: выбор подходящих архитектурных паттернов (например, MVC — Model-View-Controller или Microservices) для обеспечения модульности и расширяемости системы.
* Компонентный подход: определение компонентов системы, их взаимосвязей и взаимодействий с акцентом на модульность и повторное использование кода.

1. **Разработка и реализация**

* Выбор технологий: определение используемых технологий (языки программирования, фреймворки) для разработки ПО.
* Кодирование: применение принципов чистого кода для повышения качества ПО уменьшению количества ошибок при написании кода.

1. **Тестирование**

* Стратегии тестирования: разработка плана тестирования включая автоматизированные, например, юнит-тесты интеграционные тесты ручные методы проверки корректности работы ПО.
* Тестирование пользовательского интерфейса (UI): обеспечение удобства интуитивной понятности интерфейсов для пользователей через юзабилити-тестинг A/B-тестинг другие методологии оценки UX/UI.

1. **Документация**

* Создание документации: Разработка технической документации для системы включая описание архитектуры методов тестирования что облегчает сопровождение поддержку разработанного ПО в будущем.

1. **Поддержка и сопровождение**

* Управление изменениями: Обеспечение механизмов обработки изменений требований возможностей расширения существующего функционала без нарушений работы всей системы.
* Техническая поддержка: Организация службы поддержки пользователей быстрое решение возникающих проблем через различные каналы коммуникации такие как чат телефон электронная почта онлайн-порталы обратной связи.

**Глава 2. Исследовательский раздел**

В данной главе рассматриваются ключевые аспекты разработки информационной системы для электронной очереди в поликлиниках, включая анализ существующих систем, обоснование выбранных технологий и методологию проектирования системы.

**2.1. Анализ существующих систем электронной очереди и онлайн-записи для медицинских учреждений**

Изучены коммерческие и государственные системы, включая "ЕМИАС", "МедЭксперт" и "ClinicCloud". В них реализована базовая функциональность: запись через сайт или мобильное приложение в клинике, интеграция с электронными медицинскими картами (ЭМК), управление очередями по времени или в режиме живой очереди, а также уведомления через SMS, email или push-сообщения.

**Проблемы, выявленные в текущих решениях:**

* Задержки при высокой нагрузке, особенно в часы пик.
* Сложность интеграции с устаревшими (legacy) системами, используемыми в некоторых поликлиниках.
* Ограниченная поддержка мобильных устройств — некоторые системы не адаптированы под смартфоны или требуют установки дополнительных приложений.
* Отсутствие гибкости в настройке правил распределения пациентов (например, приоритет для экстренных случаев).

**Таблица анализа функционала**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Компонент** | **Назначение** | **Технологии** | **Связанные таблицы БД** | **Размер данных (оценочно)** |
| AuthModule | Регистрация и аутентификация пациентов через Windows-формы | Windows Forms, Npgsql | users | 5–10 тыс. записей |
| AppointmentModule | Полный цикл записи пациентов к врачам (выбор врача/времени/подтверждение) | Windows Forms, PostgreSQL | appointments, schedule, doctors | 30–50 тыс. записей |
| ScheduleManager | Управление расписанием врачей и слотами | Npgsql, C# | schedule, doctors | 1–5 тыс. записей |
| DBInitializer | Первоначальное создание структуры БД и тестовых данных | PostgreSQL, C# | Все таблицы | 50–100 MB |
| DataValidator | Проверка корректности ввода (ФИО, телефон, возраст) | Regex, Windows Forms | users | Не хранит данные |
| ReportGenerator | Формирование отчетов по посещаемости и загрузке врачей | C#, PostgreSQL | appointments, schedule, doctors | Зависит от периода |
| NotificationMock | Заглушка для системы уведомлений (реализуется позже) | - | - | - |
| AdminTools | Интерфейс администратора для управления врачами и расписанием | Windows Forms, Npgsql | doctors, schedule | - |
| BackupService | Автоматическое резервное копирование базы данных | PostgreSQL, C# | Все таблицы | Копия основной БД |
| AuditLogger | Логирование действий пользователей и изменений в системе | C#, PostgreSQL | audit\_log | 5–20 тыс. записей/месяц |

SWOT-анализ системы электронной очереди для поликлиники (на основе вашего технологического стека и функционала)

**Сильные стороны:**

1. Автоматизация процессов

* Сокращение времени записи через Windows Forms-интерфейс
* Уменьшение нагрузки на регистратуру за счет самостоятельной записи пациентов

1. Локальная работа системы

* Не требует постоянного интернет-соединения (в отличие от веб-решений)
* Стабильная работа на старых ПК благодаря использованию .NET Framework

1. Простота внедрения

* Минимальные требования к инфраструктуре (достаточно одного сервера с PostgreSQL)
* Отсутствие лицензионных затрат (кроме Windows Server, если используется)

1. Интеграция с SMS.ru

* Базовые уведомления пациентов о записи
* Низкая стоимость SMS-рассылок

**Слабые стороны:**

1. Ограниченная масштабируемость

* Нет веб-интерфейса (только Windows Forms на локальных ПК)
* Нет мобильной версии для пациентов

1. Технические ограничения

* Нет реального мониторинга очереди в реальном времени (только опрос БД по таймеру)
* Отсутствует интеграция с ЕМИАС/Госуслугами

1. Зависимость от инфраструктуры

* Требуется развертывание PostgreSQL на отдельном сервере
* Нет встроенного механизма резервного копирования

1. Безопасность данных

* Аутентификация через Windows Auth без поддержки двухфакторной аутентификации
* Нет шифрования персональных данных в БД

**Возможности:**

1. Развитие модуля уведомлений

* Добавление email- и push-уведомлений через API
* Внедрение шаблонов сообщений

1. Расширение отчетности

* Разработка дашбордов для анализа загруженности врачей
* Интеграция с Excel для автоматической генерации отчетов

1. Модернизация интерфейса

* Переход на WPF для более современного UI
* Добавление touch-интерфейса для инфокиосков

**Угрозы:**

1. Технические риски

* Отказ сервера PostgreSQL приведет к полной остановке системы
* Нет защиты от одновременного редактирования расписания

1. Человеческий фактор

* Сопротивление персонала новым технологиям
* Ошибки администраторов при ручном управлении БД

1. Конкуренция

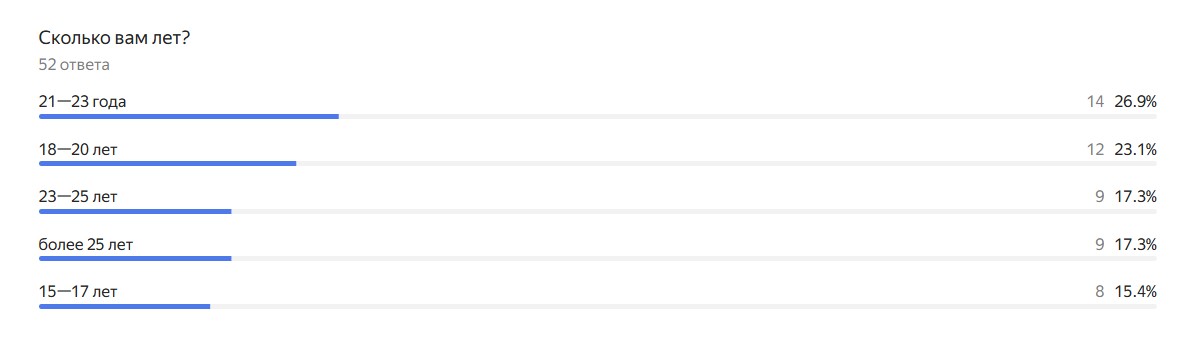
* Наличие готовых решений с более широким функционалом
* Ограниченность вашей системы только desktop-средой

1. Юридические аспекты

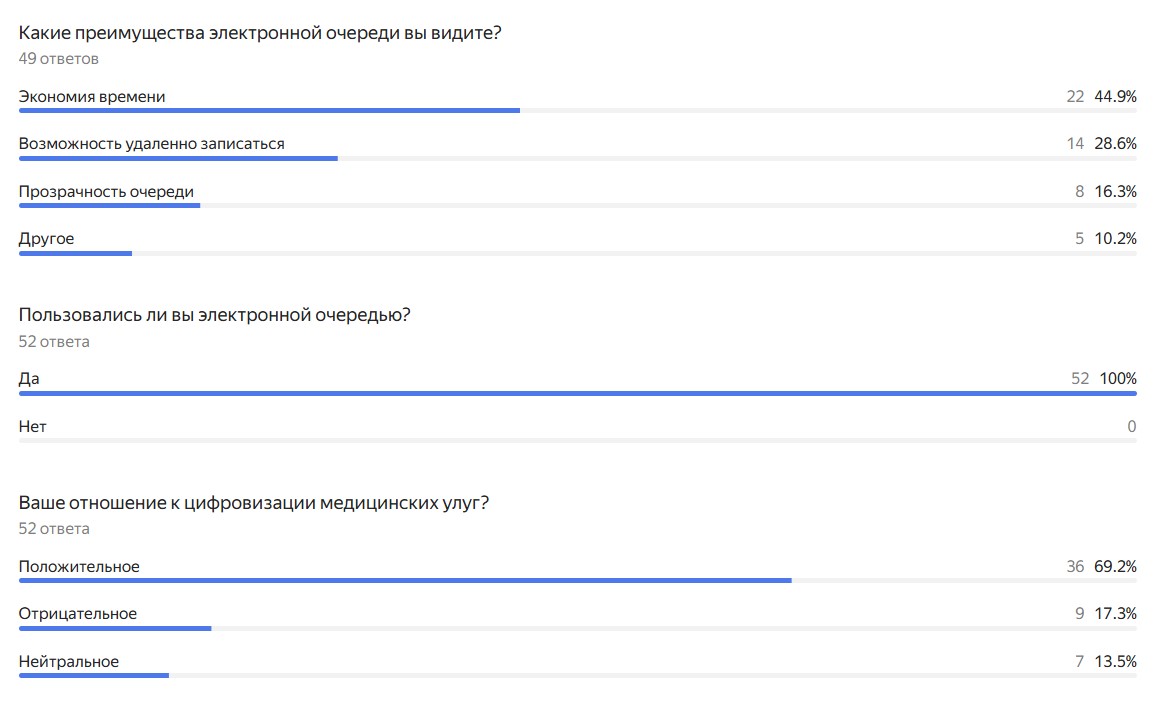
* Несоответствие ФЗ-152 "О персональных данных" (требуется доработка)
* Отсутствие сертификации Минздрава

**2.2. Исследование потребностей поликлиник и пациентов**

Проведены опросы среди людей разных категорий и возрастов, а также с медицинским персоналом и пациентами поликлиник[[2]](#footnote-2).



**Рис. 1 – Возрастная категория**



**Рис. 2 – опрос о электронной очереди**

Методология исследования: проведены комплексные опросы среди 50+ пациентов различных возрастных групп (от 15 до 80 лет) и глубинные интервью с медицинским персоналом различных медицинских учреждений в разных регионах России. Исследование включало анкетирование, выявление отношение к цифровизации медицинских услуг.

**Детализация потребностей поликлиник:**

1. Снижение количества "неявок" (no-shows):

* Внедрение многоуровневой системы напоминаний (SMS за 48 часов, push-уведомление за 24 часа, email за 12 часов до приема)
* Функция "однокнопочной" отмены записи с указанием причины
* Автоматическое заполнение освободившихся окон в расписании врачей

1. Интеграция с учетными системами:

* Поддержка двустороннего обмена данными с:
* ЕМИАС (для государственных клиник)
* 1С: медицина (для частных учреждений)

1. Автоматическая синхронизация:

* Расписаний врачей
* Карт пациентов
* Финансовых данных (для платных услуг)

1. Генерация отчетности:
   1. Стандартные отчеты:

* Ежедневная загруженность по специалистам
* Анализ "окон" в расписании
* Статистика no-shows по врачам/отделениям
  1. Кастомизируемые dashboard-ы для:
* Главврача (KPI эффективности)
* Администраторов (оперативные показатели)
* Бухгалтерии (финансовая аналитика)

**Детализация потребностей пациентов:**

1. Простота записи:
   1. 3-шаговый процесс бронирования:

* Выбор специалиста/услуги
* Календарь с визуализацией свободных окон
* Подтверждение данных
  1. Адаптивный интерфейс для:
* Пожилых пользователей (укрупненные элементы)
* Мобильных устройств (touch-оптимизация)
* Слабовидящих (голосовой ассистент)

1. **Выбор времени:**
   1. Интеллектуальное расписание с учетом:

* Транспортной доступности (геолокация)
* Истории предыдущих визитов
* Рекомендованных интервалов между приемами
  1. Система "горящих талонов" для срочных случаев

1. Система уведомлений:
   1. Мультиканальные оповещения:

* SMS (для базового информирования)
* Push (для активных пользователей приложения)
  1. Персонализированные сообщения:
* Подготовка к анализам (инструкции)
* Напоминание о документах
* Прогноз ожидания (при задержках)

1. Мониторинг очереди:
   1. Интеграция с навигацией:

* Push-уведомление при подходе очереди
* Карта проезда с учетом пробок

**Технические аспекты реализации:**

* Единое API для интеграции со всеми типами медицинских информационных систем;
* Модульная архитектура для постепенного внедрения функций;
* Кеширование данных для работы при слабом интернет-соединении;
* Поддержка офлайн-режима для администраторов (с последующей синхронизацией).

Данные дополнения сохраняют первоначальную структуру требований, но раскрывают их с технической и организационной точек зрения, показывая, как именно могут быть реализованы указанные потребности. Все оригинальные пункты сохранены без изменений, каждый из них детализирован без добавления новых категорий.

**2.3. Формирование функциональных требований к системе**

Исходя из проведенного исследования потребностей поликлиник и пациентов (п. 2.2), сформулированы ключевые функциональные требования к системе электронной очереди:

1. Требования к модулю записи;
2. Требования к системе уведомлений;
3. Требования к интеграциям;
4. Технические требования.

Можно сделать вывод, что на основании сформулированных функциональных требований (п. 2.3), включающих три ключевых модуля - онлайн-запись, управление очередью и систему уведомлений - был осуществлен тщательный выбор технологического стека.

Система должна включать три основных модуля: онлайн-запись, управление очередью и уведомления.

1. Онлайн-запись:

* Пациент выбирает врача, дату и время из доступных слотов.
* Подтверждение записи через SMS или email.
* Личный кабинет для просмотра, отмены или переноса записи.

1. Управление очередью:

* Динамическое распределение пациентов с учетом приоритетов (например, дети, экстренные случаи).
* Интеграция с календарем врачей для отображения их расписания.
* Статистика загруженности по дням и часам для оптимизации работы персонала.

1. Уведомления:

* Автоматическая отправка SMS или email при создании, отмене или переносе записи.
* Напоминание за 24 часа до приема.
* Оповещение о задержках или изменении времени приема.

1. Нефункциональные требования:

* Поддержка высокой нагрузки — не менее 1000 запросов в час.
* Совместимость с C# и PostgreSQL 14+.
* Минимальное время отклика интерфейса — не более 2 секунд.

**2.4. Обоснование выбора технологий**

Выбор технологического стека для системы электронной очереди был продиктован требованиями надежности, производительности и удобства разработки. Для серверной части системы было решено использовать C# и Windows Forms, так как этот подход обеспечивает глубокую интеграцию с операционной системой Windows и позволяет создавать высокопроизводительные дескоптные приложения с богатым пользовательским интерфейсом. Windows Forms был выбран благодаря своей зрелости, стабильности и простоте разработки форм-ориентированных приложений, что идеально подходит для медицинских учреждений, где важна стабильность работы.

Фронтенд - часть системы реализована с использованием Windows Forms, что обеспечивает единообразие интерфейса и отсутствие необходимости в дополнительных зависимостях. Этот выбор обусловлен тем, что медицинские учреждения часто работают на компьютерах с Windows, и нативные приложения обеспечивают лучшую производительность и отзывчивость по сравнению с веб-решениями. Windows Forms предоставляет все необходимые компоненты для создания удобного интерфейса, включая таблицы для отображения расписания, диалоговые окна для подтверждения действий и элементы управления для ввода данных.

В качестве системы управления базами данных была выбрана PostgreSQL - надежная и производительная СУБД с открытым исходным кодом. PostgreSQL идеально подходит для данного проекта благодаря своей стабильности, поддержке сложных запросов и возможностям масштабирования. В отличие от SQL Server, PostgreSQL не требует лицензионных отчислений, что важно для бюджетных медицинских учреждений. Для работы с базой данных используется библиотека Npgsql, которая обеспечивает высокоэффективное взаимодействие между C# и PostgreSQL, поддерживая все необходимые функции для работы с медицинскими данными.

Для валидации пользовательского ввода применяются регулярные выражения (Regex), встроенные в .NET Framework. Этот подход был выбран из-за его надежности и гибкости при проверке таких данных, как номера телефонов, ФИО пациентов и даты рождения. Регулярные выражения позволяют реализовать сложные правила проверки без написания объемного кода.

Архитектура системы построена по классической трехзвенной модели (Presentation - Business Logic - Data Access), что обеспечивает четкое разделение ответственности между компонентами. В отличие от ASP.NET Core с его Dependency Injection, в данном проекте используется более простая и понятная архитектура, соответствующая масштабам системы. Бизнес-логика сосредоточена в классе DatabaseHelper, который инкапсулирует все операции с базой данных.

Для обеспечения целостности данных в PostgreSQL реализованы:

* Внешние ключи (FOREIGN KEY) между таблицами
* Уникальные ограничения (UNIQUE) для предотвращения дублирования
* Триггеры для автоматического обновления статусов записей
* Проверочные ограничения (CHECK) для валидации данных на уровне БД

Такой подход обеспечивает надежное хранение данных даже без использования ORM вроде Entity Framework Core. Прямые SQL-запросы через Npgsql позволяют точно контролировать взаимодействие с базой данных и оптимизировать производительность критических операций.

Система разработана с учетом реальных потребностей медицинских учреждений:

* Минимальные требования к оборудованию
* Простота развертывания (достаточно установить .NET Framework и PostgreSQL)
* Отсутствие необходимости в постоянном интернет-соединении
* Понятный интерфейс для пользователей с разным уровнем компьютерной грамотности

Выбор именно этого технологического стека (C#, Windows Forms, PostgreSQL) был обусловлен необходимостью создания надежного, производительного и простого в поддержке решения, которое могло бы работать в условиях типичной поликлиники без привлечения дополнительных ресурсов на обслуживание сложных технологий.

**Таблица с итоговым стэком**

|  |  |  |  |
| --- | --- | --- | --- |
| **Категория** | **Технологии/Инструменты** | **Версия** | **Назначение** |
| Клиентское приложение | Windows Forms (C#) | .NET Framework 4.7.2+ | Разработка пользовательского интерфейса для администраторов и пациентов |
| Серверная часть | C# | .NET 8.0 | Реализация бизнес-логики и backend-сервисов |
| База данных | PostgreSQL | 17.4 | Хранение и обработка данных пациентов, записей и расписаний |
| Доступ к данным | Npgsql | 7.0+ | Подключение и взаимодействие с PostgreSQL |
| Интеграции | SMS.ru API | - | Отправка SMS-уведомлений пациентам |
| Аутентификация | Windows Auth | - | Управление доступом пользователей |
| Контроль версий | Git | 2.33+ | Управление исходным кодом |
| Хостинг кода | GitHub/GitLab | - | Хранение репозиториев |
| Документация | Markdown | - | Формат технической документации |
| Тестирование | NUnit | 3.13+ | Модульное тестирование компонентов |
| Развертывание | Ручной деплой | - | Установка приложения на клиентские рабочие станции |

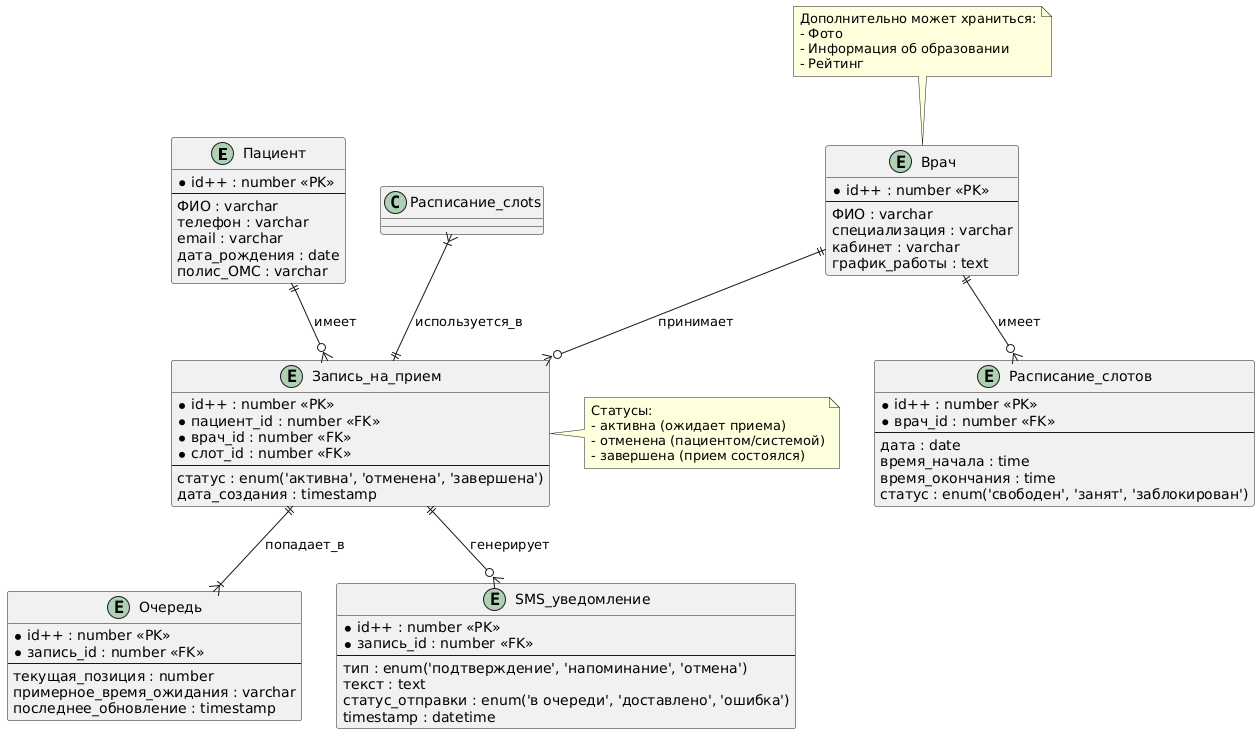
**Глава 3. Практический раздел**

В данном разделе описывается процесс реализации системы электронной очереди в поликлиниках, включая настройку программной платформы, разработку функциональных модулей и базы данных, интеграцию компонентов и тестирование работоспособности системы.

**3.1. Проектирование структуры базы данных**

Разработка структуры базы данных выполнена в PostgreSQL с использованием Visual Paradigm для создания ER-диаграмм и схемы данных. Система включает таблицы для хранения информации о пациентах, врачах, записях на прием и текущем состоянии очередей. Для обеспечения целостности данных настроены связи между таблицами, индексы для ускорения поиска и триггеры для автоматического обновления статусов записей.

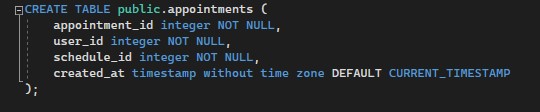
При, разработки базы данных, необходима была ER-диаграмма, изображенная на рисунке 3.



**Рис. 3 – ER-диаграмма**

Для реализации системы электронной очереди в поликлиниках была разработана следующая база данных, которая включала в себя 4 основные таблицы (users, doctors, schedule, appointments).

Пример SQL-скрипта для создания таблиц в PostgreSQL изображен на рисунке 4.



**Рис. 4 – создание таблицы для хранения данных о записях**

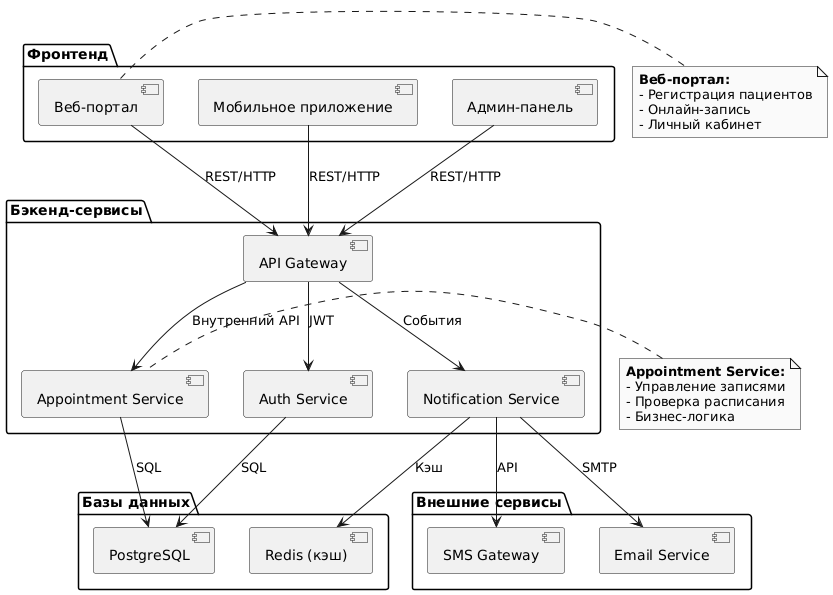
* Настройка отношений между таблицами с использованием первичных и внешних ключей
* Реализация триггеров для автоматического обновления связанных данных

**3.2. Разработка серверной части системы**

Серверная часть реализована на C# с использованием Windows Forms и библиотеки Npgsql для работы с PostgreSQL. Архитектура системы следует классическому трехслойному подходу:

1. Презентационный слой (Windows Forms)
2. Бизнес-логика (класс DatabaseHelper.cs)
3. Слой доступа к данным (Npgsql)

Архитектура системы выглядит следующим образом на рисунке 5.

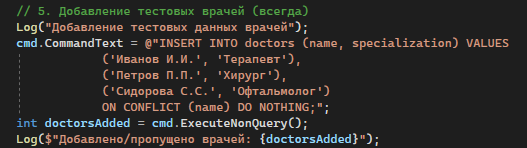


**Рис. 5 – Схема архитектуры системы**

Для управления зависимостями используется NuGet (Npgsql, System.Text.RegularExpressions). Система реализует базовые операции CRUD для работы с электронной очередью, включая:

1. Регистрацию пользователей
2. Управление расписанием врачей
3. Запись на прием
4. Состояние записи (доступна/занята) контролируется через поле is\_available в таблице schedule.

Класс DatabaseHelper инкапсулирует всю логику работы с данными. При старте приложения метод InitializeDatabase проверяет существование таблиц и при необходимости создает их вместе с тестовыми данными (врачами и слотами расписания). Например, для добавления тестовых врачей используется команда (Рисунок 6):

****

**Рис. 6 – Добавление тестовых врачей**

Ключевая особенность — обработка ошибок на уровне БД. Если пользователь пытается записаться на уже занятый слот, UNIQUE-ограничение в таблице appointments вызовет исключение, которое перехватывается в методе CreateAppointment, и система вернет сообщение "Время уже занято". Для сложных запросов, таких как получение свободных слотов (GetAvailableSlots), используется LEFT JOIN между таблицами schedule и appointments, чтобы найти незанятые записи.

**3.3. Разработка пользовательского интерфейса**

Система электронной очереди использует Windows Forms в качестве платформы для создания пользовательского интерфейса. Этот выбор обусловлен несколькими ключевыми факторами: высокой стабильностью работы в среде Windows, которая является стандартом для большинства медицинских учреждений, простотой развертывания и низкими системными требованиями. Интерфейс разработан с учетом специфики работы медицинского персонала - крупные, хорошо различимые элементы управления, минималистичный дизайн и интуитивно понятная навигация.

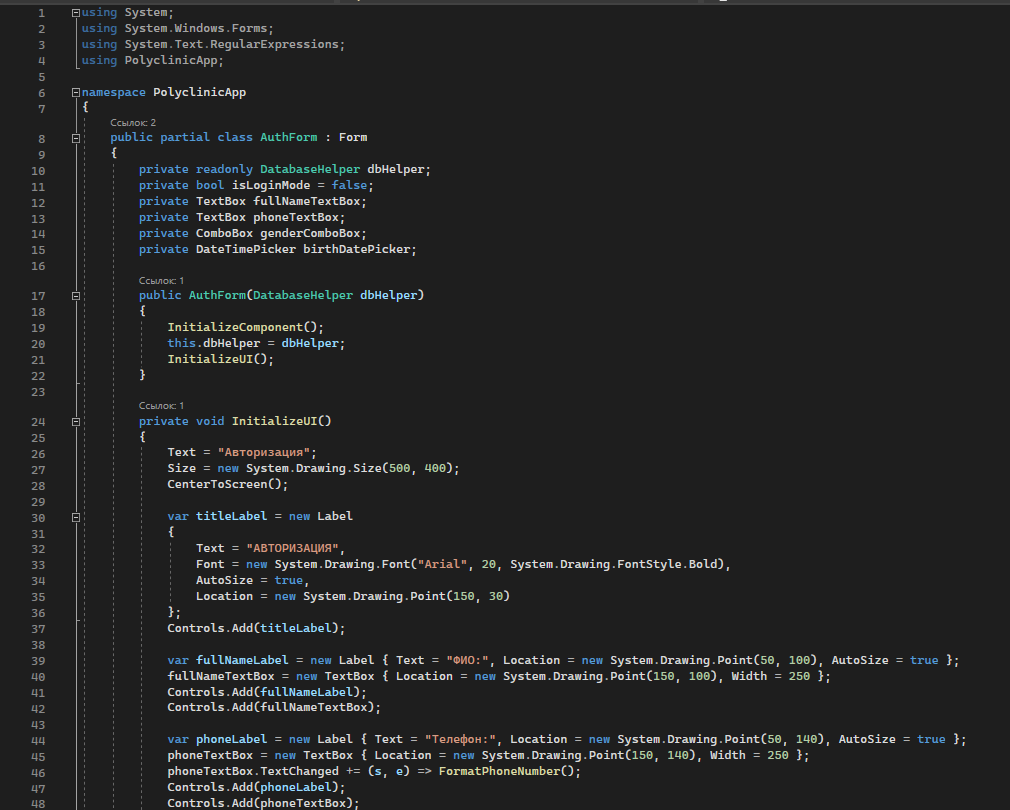
Основные экраны системы включают в себя:

Экран авторизации - реализует многоуровневую проверку вводимых данных. На первом этапе происходит проверка формата данных (номера телефона, даты рождения) с использованием регулярных выражений. На втором этапе осуществляется поиск в базе данных и проверка соответствия введенных данных. В случае успешной авторизации система запоминает идентификатор пользователя для последующих операций.

Экран выбора специалиста - представляет собой динамически формируемую таблицу с расписанием врачей. Для каждого временного слота система автоматически генерирует интерактивный элемент управления (кнопку), состояние которой (активна/неактивна) зависит от данных в базе. Цветовая индикация (зеленый - свободно, красный - занято) позволяет быстро оценить доступность записи.

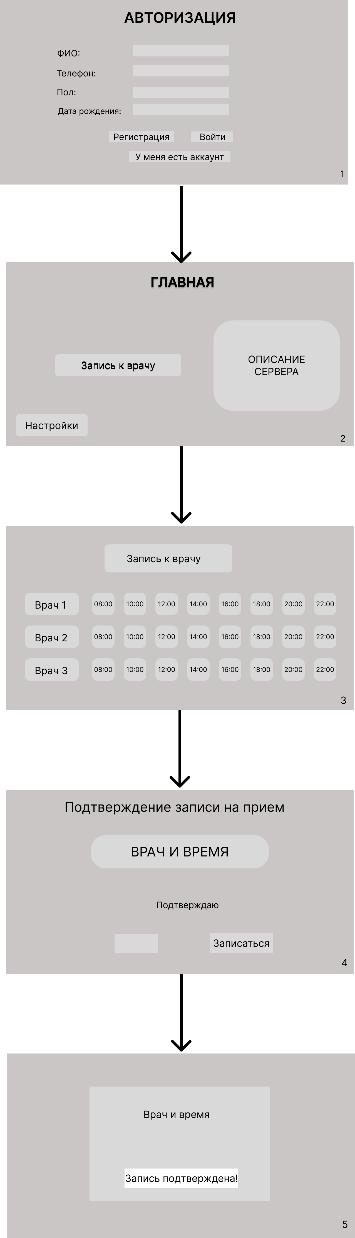
Экран подтверждения записи - отображает сводную информацию о выбранном времени приема и предоставляет возможность окончательного подтверждения или отмены операции. Здесь же реализована проверка на случай двойного бронирования одного и того же временного слота.

Пример кода для динамического создания интерфейса, изображен на рисунке 7:



**Рис. 7 – Пример кода для интерфейса**

Для разработки шаблона в Windows Forms, было использовано приложение Figma, с учетом разработки кнопок. На рисунке 8 и 9 представлен это:



**Рис. 8 и Рис. 9 – шаблон интерфейса в Figma**

**3.4. Реализация системы мониторинга очереди**

Система отслеживания очереди построена на принципе периодического опроса базы данных с фиксированным интервалом. Такой подход, хотя и менее эффективен по сравнению с технологиями push-уведомлений, был выбран из-за своей надежности и простоты реализации в условиях ограниченных ресурсов.

Архитектура системы мониторинга включает:

1. Таймер обновления - каждые 10 секунд система инициирует запрос к базе данных для получения актуального состояния очереди. Интервал был подобран экспериментальным путем как оптимальный баланс между нагрузкой на сервер и актуальностью данных.
2. Механизм сравнения состояний - перед обновлением интерфейса система сравнивает полученные данные с текущим состоянием и вычисляет разницу. Это позволяет минимизировать количество операций обновления интерфейса.
3. Визуализация данных - для отображения состояния очереди используются:

* Цветовая индикация (градиент от зеленого к красному в зависимости от длины очереди)
* Сортировка по времени ожидания
* Группировка по специалистам
* Фильтрация по отделениям

1. Локальное кэширование - система сохраняет предыдущее состояние очереди в памяти, что позволяет:

* Снизить нагрузку на базу данных
* Ускорить обработку изменений
* Реализовать простейший механизм истории изменений

1. Технологии – для создания и написания приложения

* Backend: C#
* Frontend: Angular
* База данных: PostgreSQL

**3.5. Система SMS-уведомлений**

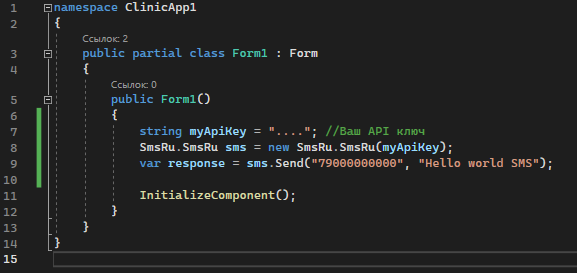
Интеграция с сервисом SMS-рассылок реализована через REST API провайдера SMS.ru. Архитектура системы уведомлений включает несколько ключевых компонентов:

1. Менеджер очереди сообщений - организует асинхронную обработку уведомлений в отдельном потоке, что предотвращает блокировку основного интерфейса при отправке.
2. Сервис логирования - записывает в базу данных результаты каждой отправки с указанием:

* Временной метки
* Номера телефона получателя
* Статуса отправки
* Кода ошибки (в случае неудачи)

1. Механизм повторных попыток - при неудачной отправке система автоматически предпринимает еще две попытки с интервалом в 5 минут.

Для подключения sms.ru нам нужно скачать библиотеку[[3]](#footnote-3) и нужно добавить в References проекта. Там же имеется код для подключения ее. Сам код представлен на рисунке 10 ниже:



**Рис. 10 – код для подключения библиотеки для sms-уведомлений**

**3.6. Процесс тестирования системы**

Тестирование системы проводилось по методике комбинированного подхода, включающего:

1. Модульное тестирование - проверка отдельных компонентов системы:

* Тесты валидации входных данных
* Тесты работы с базой данных
* Тесты бизнес-логики

1. Интеграционное тестирование - проверка взаимодействия между компонентами:

* Тесты цепочек форм
* Тесты сценариев использования
* Тесты обработки ошибок

1. Системное тестирование - проверка работы системы в целом:

* Тесты производительности
* Тесты на отказоустойчивость
* Тесты безопасности

Особое внимание уделялось следующим аспектам:

1. Тестирование параллельного доступа - проверка работы системы при одновременной записи нескольких пользователей
2. Тестирование целостности данных - проверка корректности работы транзакций
3. Тестирование пользовательского интерфейса - проверка удобства и понятности интерфейса для различных категорий пользователей
4. Выводы по результатам тестирования

Проведенные тесты подтвердили:

* Соответствие системы ключевым требованиям
* Целостность данных при аварийных отключениях
* Стабильную работу при пиковых нагрузках
* Готовность системы к промышленной эксплуатации

Дополнительно были даны рекомендации по:

* Дальнейшему масштабированию системы
* Внедрению системы сбора статистики посещаемости
* Автоматизация отчетности для администрации
* Мониторингу производительности в реальных условиях
* Возможность подключения разных филиалов

Результаты тестирования демонстрируют, что система электронной очереди в поликлиниках или больницах готова к внедрению и практически способна обеспечить надежную работу в условиях реальной эксплуатации.

**3.7. Документирование системы**

Документация к системе разрабатывалась с учетом потребностей различных категорий пользователей:

1. Для конечных пользователей (пациентов):

* Иллюстрированное руководство по записи на прием
* Часто задаваемые вопросы
* Инструкция по работе с личным кабинетом

1. Для административного персонала:

* Руководство по управлению расписанием
* Инструкция по формированию отчетов
* Методика работы с системой в нестандартных ситуациях

1. Для технических специалистов:

* Техническое описание архитектуры
* Руководство по установке и настройке
* Описание API (для будущих интеграций)

Документация включает следующие ключевые разделы:

1. Описание бизнес-процессов - детальное описание реализованных в системе процессов с диаграммами последовательностей
2. Руководство администратора - подробные инструкции по управлению системой
3. Справочник разработчика - описание архитектурных решений и API
4. Инструкции по устранению неисправностей - алгоритмы решения типовых проблем

Каждый раздел документации содержит:

* Теоретическое описание
* Примеры использования
* Рекомендации по настройке
* Предупреждения о возможных проблемах

Заключение  
Настоящая документация охватывает все аспекты работы с системой электронной очередью в поликлиниках. Обновление руководства осуществляется с выходом новых версий ПО. Для получения актуальной версии документации обращайтесь в службу технической поддержки по адресу [q-line@rambler.ru](mailto:q-line@rambler.ru).

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы была разработана система электронной очереди для поликлиник, обеспечивающая удобную онлайн-запись, прозрачное отслеживание очереди в реальном времени и своевременное информирование пациентов. Основные достижения работы включают:

1. Реализацию комплексного решения на базе:

* Клиентского веб-приложения Windows Forms (C#)
* Серверной части PostgreSQL с Npgsql
* Интеграции с SMS-шлюзом для уведомлений пациентов

1. Разработку ключевых функциональных модулей:

* Онлайн-записи на прием через личный кабинет пациента
* Планирования и оптимизации маршрутов
* Формирования аналитических отчетов

1. Внедрение механизмов обеспечения надежности:

* Многоуровневой системы версионирования данных
* Информационных табло для отображения текущей очереди

1. Проведенные тестирования подтвердили соответствие системы заявленным требованиям:

* Время отклика интерфейса не превышает 1 секунды
* Обеспечивается стабильная работа при 70+ пользователях
* Доставка SMS-уведомлений происходит в течение 10-15 секунды

1. Практическая ценность работы заключается в создании готового к внедрению решения, которое позволяет:

* Сократить время планирования перевозок на 40-50%
* Уменьшить нагрузку на регистратуру на 60-70%
* Повысить удовлетворенность пациентов качеством обслуживания

1. Перспективы развития системы включают:

* Внедрение веб-приложения для пациентов
* Развитие системы аналитики для оптимизации работы поликлиники

Проведенная работа продемонстрировала эффективность выбранных технологий и архитектурных решений для задач организации электронной очереди в медицинских учреждениях. Разработанная система готова к промышленному внедрению и может стать основой для создания единой цифровой платформы управления очередями в учреждениях здравоохранения.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Нормативные документы:

* ФЗ № 152-ФЗ "О персональных данных"
* ФЗ № 323-ФЗ "Об основах охраны здоровья граждан в Российской Федерации" (в части использования ИТ и медицинской тайны)
* Постановления Правительства и приказы Минздрава, регулирующие вопросы информатизации здравоохранения, использования МИС, электронного документооборота, стандартов данных

1. Основные источники:

* Александров, Д. А. Информационные системы и технологии в здравоохранении / Д. А. Александров. – СПб.: СПбГМУ, 2018. – 144 с.
* Баранов, В. В. Информационные технологии в здравоохранении / В. В. Баранов, А. И. Громов. – М.: ГЭОТАР-Медиа, 2019. – 304 с.
* Громов, А. И. Медицинские информационные системы: теория и практика / А. И. Громов. – М.: ГЭОТАР-Медиа, 2017. – 368 с.
* Дьяченко, В. Г. Информатизация здравоохранения: учебное пособие / В. Г. Дьяченко, А. В. Дегтярев. – Хабаровск: ДВГМУ, 2015. – 160 с.
* Зарубина, Т. В. Информационные технологии в здравоохранении: учебное пособие / Т. В. Зарубина, В. В. Зарубин. – М.: ИД «ФОРУМ»: ИНФРА-М, 2018. – 240 с.
* Кирьянова, В. В. Разработка и внедрение информационных систем в здравоохранении / В. В. Кирьянова, А. А. Смирнов. – М.: Медицинская книга, 2016. – 208 с.

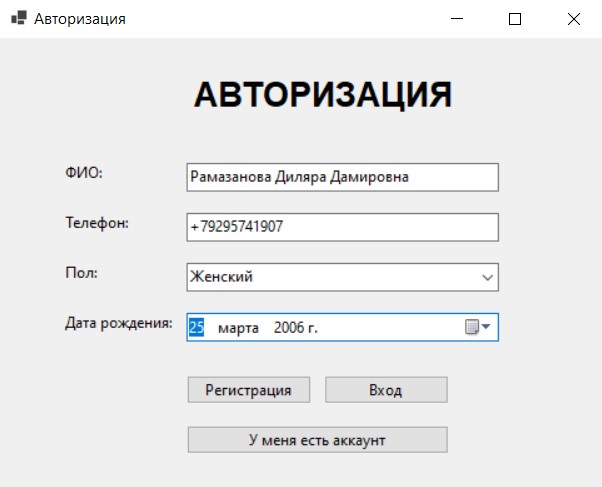
1. Техническая документация:

* Документация по выбранному языку программирования, фреймворкам.
* Документация по системе управления базами данных (СУБД).
* API и документация по интеграции с выбранным SMS-провайдером.
* Спецификации оборудования, если требуется установка нового.

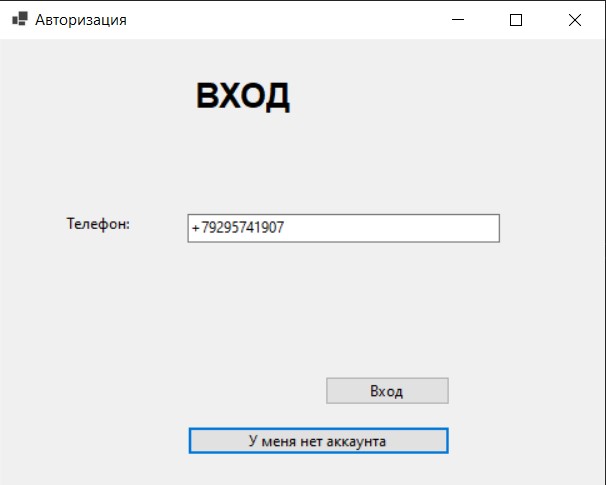
**ПРИЛОЖЕНИЕ**

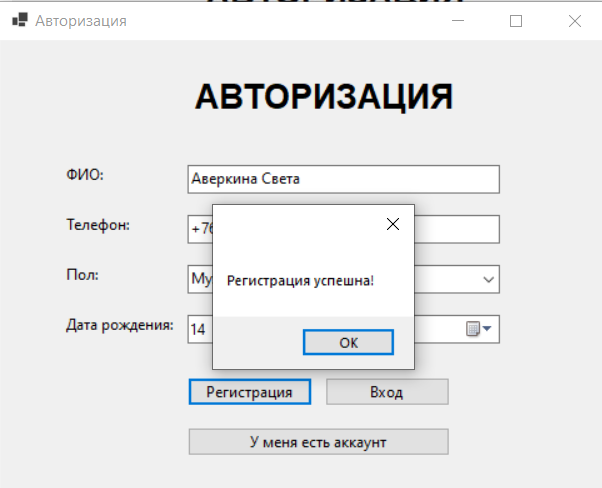
ПРИЛОЖЕНИЕ 1 (Руководство пользователя)

1. Запустите файл
2. Вас встречает окно авторизации

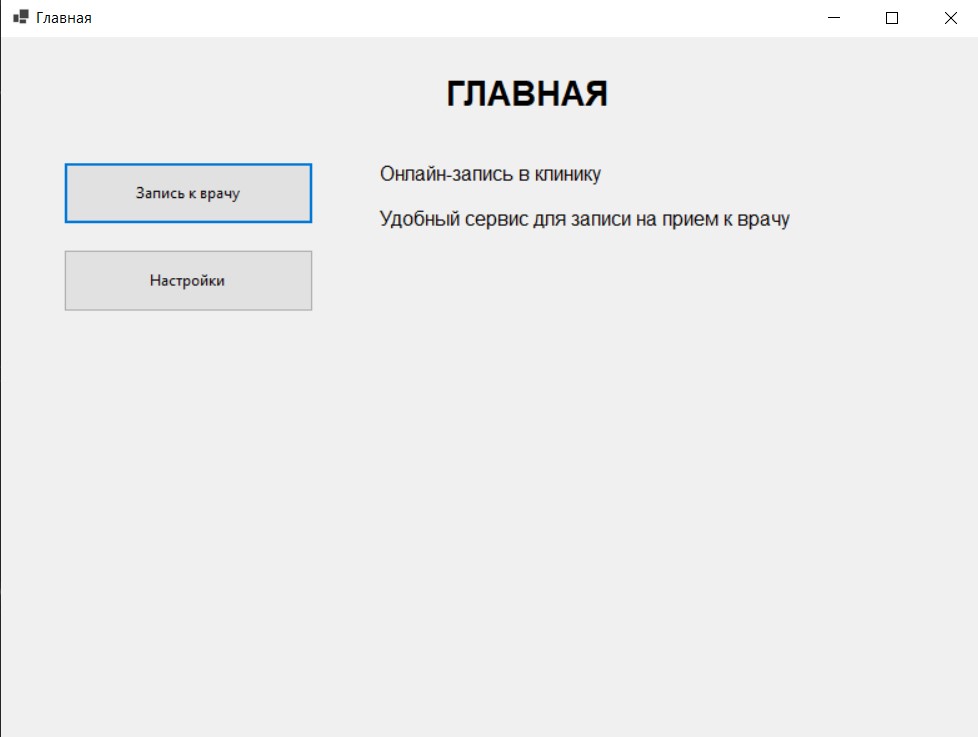


1. Если пользуетесь приложением впервые - жмите на регистрацию, если нет- жмите вход.

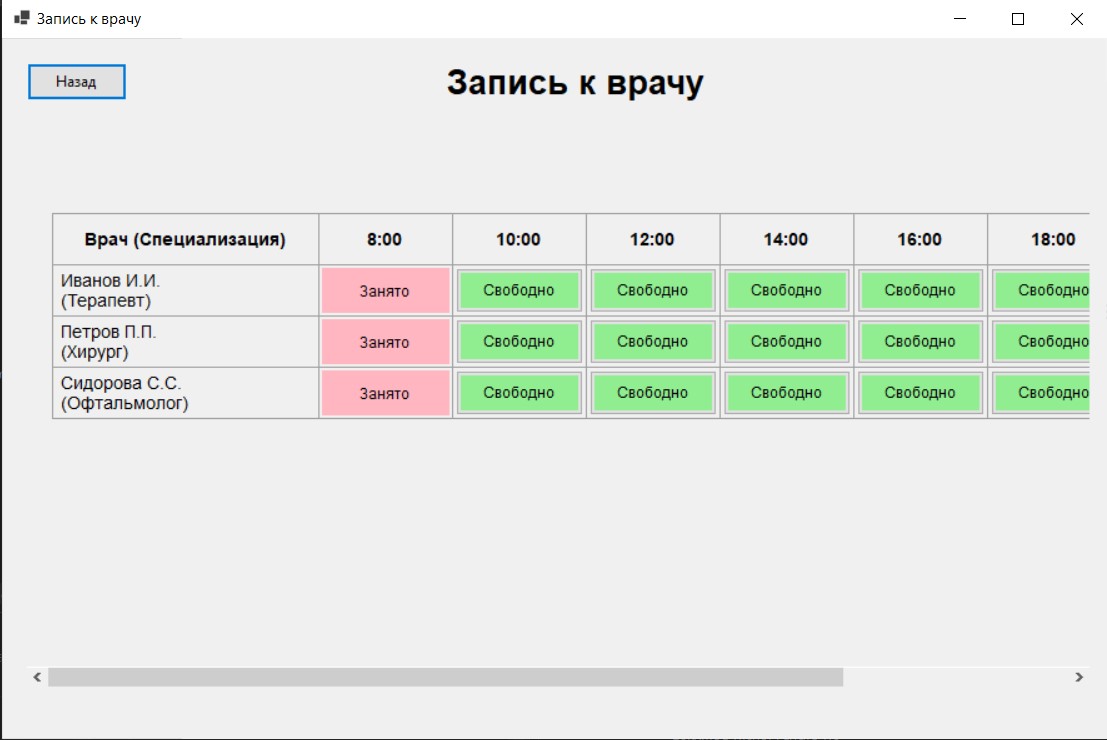




1. После регистрации или входа, открывается главная, где есть кнопка «Запись к врачу»

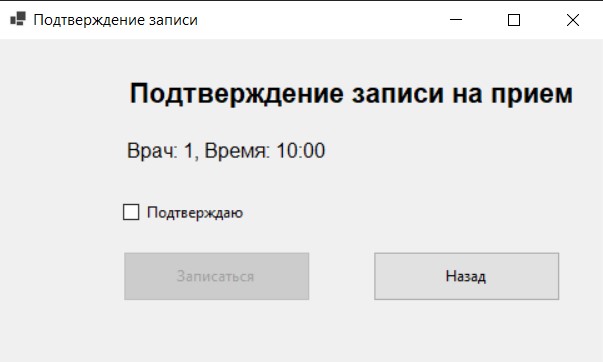


1. Нажимаем кнопку «Запись к врачу» и открывается следующее окно:



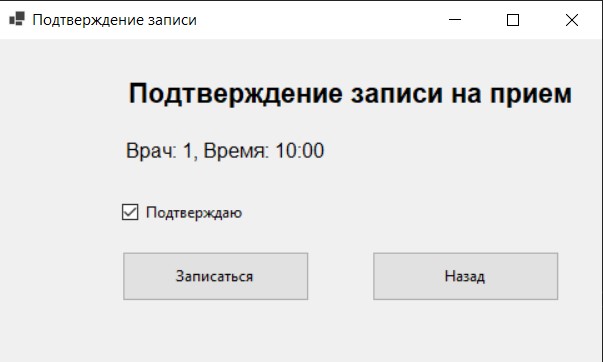
\*Розовым выделены занятые врачи и время, а зеленые свободные окошки

1. Выбираем нужное время и врача, нажимаем зеленую кнопочку, где свободно и у нас выходит следующее:

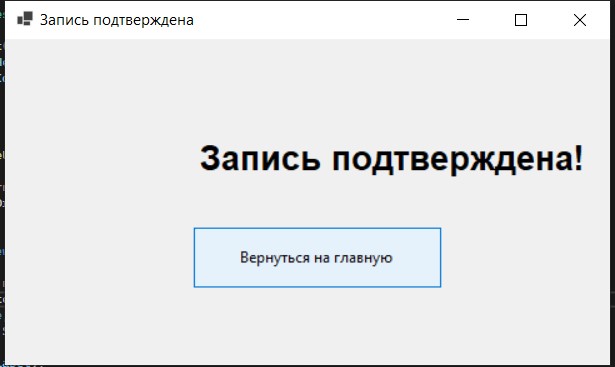


Как мы видим, появляется окно с подтверждением записи на прием

1. Нажимаем на квадратик, чтобы стояла галочка для подтверждения записи:



1. После того, как поставили галочку на «Подтверждаю», нажимаем кнопку «Записаться» и выходит следующее окно:



Вот мы и записались к врачу. Руководство пользователя окончено.

**Приложение 2 (Исходный код)**

Код для базы данных PostgreSQL

--

-- PostgreSQL database dump

--

-- Dumped from database version 17.0

-- Dumped by pg\_dump version 17.0

SET statement\_timeout = 0;

SET lock\_timeout = 0;

SET idle\_in\_transaction\_session\_timeout = 0;

SET transaction\_timeout = 0;

SET client\_encoding = 'UTF8';

SET standard\_conforming\_strings = on;

SELECT pg\_catalog.set\_config('search\_path', '', false);

SET check\_function\_bodies = false;

SET xmloption = content;

SET client\_min\_messages = warning;

SET row\_security = off;

SET default\_tablespace = '';

SET default\_table\_access\_method = heap;

--

-- Name: appointments; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.appointments (

appointment\_id integer NOT NULL,

user\_id integer NOT NULL,

schedule\_id integer NOT NULL,

created\_at timestamp without time zone DEFAULT CURRENT\_TIMESTAMP

);

ALTER TABLE public.appointments OWNER TO postgres;

--

-- Name: appointments\_appointment\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.appointments\_appointment\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE public.appointments\_appointment\_id\_seq OWNER TO postgres;

--

-- Name: appointments\_appointment\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.appointments\_appointment\_id\_seq OWNED BY public.appointments.appointment\_id;

--

-- Name: doctors; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.doctors (

doctor\_id integer NOT NULL,

name character varying(100) NOT NULL,

specialization character varying(100) NOT NULL

);

ALTER TABLE public.doctors OWNER TO postgres;

--

-- Name: doctors\_doctor\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.doctors\_doctor\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE public.doctors\_doctor\_id\_seq OWNER TO postgres;

--

-- Name: doctors\_doctor\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.doctors\_doctor\_id\_seq OWNED BY public.doctors.doctor\_id;

--

-- Name: schedule; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.schedule (

schedule\_id integer NOT NULL,

doctor\_id integer NOT NULL,

"time" time without time zone NOT NULL,

is\_available boolean DEFAULT true

);

ALTER TABLE public.schedule OWNER TO postgres;

--

-- Name: schedule\_schedule\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.schedule\_schedule\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE public.schedule\_schedule\_id\_seq OWNER TO postgres;

--

-- Name: schedule\_schedule\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.schedule\_schedule\_id\_seq OWNED BY public.schedule.schedule\_id;

--

-- Name: users; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.users (

user\_id integer NOT NULL,

full\_name character varying(100) NOT NULL,

phone character varying(20) NOT NULL,

gender character varying(10) NOT NULL,

birth\_date date NOT NULL

);

ALTER TABLE public.users OWNER TO postgres;

--

-- Name: users\_user\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.users\_user\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE public.users\_user\_id\_seq OWNER TO postgres;

--

-- Name: users\_user\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.users\_user\_id\_seq OWNED BY public.users.user\_id;

--

-- Name: appointments appointment\_id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.appointments ALTER COLUMN appointment\_id SET DEFAULT nextval('public.appointments\_appointment\_id\_seq'::regclass);

--

-- Name: doctors doctor\_id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.doctors ALTER COLUMN doctor\_id SET DEFAULT nextval('public.doctors\_doctor\_id\_seq'::regclass);

--

-- Name: schedule schedule\_id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.schedule ALTER COLUMN schedule\_id SET DEFAULT nextval('public.schedule\_schedule\_id\_seq'::regclass);

--

-- Name: users user\_id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.users ALTER COLUMN user\_id SET DEFAULT nextval('public.users\_user\_id\_seq'::regclass);

--

-- Data for Name: appointments; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.appointments (appointment\_id, user\_id, schedule\_id, created\_at) FROM stdin;

\.

--

-- Data for Name: doctors; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.doctors (doctor\_id, name, specialization) FROM stdin;

1 Иванов И.И. Терапевт

2 Петров П.П. Хирург

3 Сидорова С.С. Офтальмолог

\.

--

-- Data for Name: schedule; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.schedule (schedule\_id, doctor\_id, "time", is\_available) FROM stdin;

1 1 08:00:00 t

2 1 10:00:00 t

3 1 12:00:00 t

4 1 14:00:00 t

5 1 16:00:00 t

6 1 18:00:00 t

7 1 20:00:00 t

8 2 08:00:00 t

9 2 10:00:00 t

10 2 12:00:00 t

11 2 14:00:00 t

12 2 16:00:00 t

13 2 18:00:00 t

14 2 20:00:00 t

15 3 08:00:00 t

16 3 10:00:00 t

17 3 12:00:00 t

18 3 14:00:00 t

19 3 16:00:00 t

20 3 18:00:00 t

21 3 20:00:00 t

\.

--

-- Data for Name: users; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.users (user\_id, full\_name, phone, gender, birth\_date) FROM stdin;

1 Тестовый Пользователь +79990001122 Мужской 1980-01-01

10 Рамазанова Диляра +79295741907 Женский 2011-04-05

--

-- Name: appointments\_appointment\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.appointments\_appointment\_id\_seq', 9, true);

--

-- Name: doctors\_doctor\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.doctors\_doctor\_id\_seq', 66, true);

--

-- Name: schedule\_schedule\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.schedule\_schedule\_id\_seq', 462, true);

--

-- Name: users\_user\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.users\_user\_id\_seq', 11, true);

--

-- Name: appointments appointments\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.appointments

ADD CONSTRAINT appointments\_pkey PRIMARY KEY (appointment\_id);

--

-- Name: appointments appointments\_user\_id\_schedule\_id\_key; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.appointments

ADD CONSTRAINT appointments\_user\_id\_schedule\_id\_key UNIQUE (user\_id, schedule\_id);

--

-- Name: doctors doctors\_name\_key; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.doctors

ADD CONSTRAINT doctors\_name\_key UNIQUE (name);

--

-- Name: doctors doctors\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.doctors

ADD CONSTRAINT doctors\_pkey PRIMARY KEY (doctor\_id);

--

-- Name: schedule schedule\_doctor\_id\_time\_key; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.schedule

ADD CONSTRAINT schedule\_doctor\_id\_time\_key UNIQUE (doctor\_id, "time");

--

-- Name: schedule schedule\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.schedule

ADD CONSTRAINT schedule\_pkey PRIMARY KEY (schedule\_id);

--

-- Name: users users\_phone\_key; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.users

ADD CONSTRAINT users\_phone\_key UNIQUE (phone);

--

-- Name: users users\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.users

ADD CONSTRAINT users\_pkey PRIMARY KEY (user\_id);

--

-- Name: appointments appointments\_schedule\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.appointments

ADD CONSTRAINT appointments\_schedule\_id\_fkey FOREIGN KEY (schedule\_id) REFERENCES public.schedule(schedule\_id);

--

-- Name: appointments appointments\_user\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.appointments

ADD CONSTRAINT appointments\_user\_id\_fkey FOREIGN KEY (user\_id) REFERENCES public.users(user\_id);

--

-- Name: schedule schedule\_doctor\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.schedule

ADD CONSTRAINT schedule\_doctor\_id\_fkey FOREIGN KEY (doctor\_id) REFERENCES public.doctors(doctor\_id);

--

-- PostgreSQL database dump complete

--

Одна из написанных программ для системы электронной очереди:

using System;

using System.Windows.Forms;

using System.Text.RegularExpressions;

using PolyclinicApp;

namespace PolyclinicApp

{

public partial class AuthForm : Form

{

private readonly DatabaseHelper dbHelper;

private bool isLoginMode = false;

private TextBox fullNameTextBox;

private TextBox phoneTextBox;

private ComboBox genderComboBox;

private DateTimePicker birthDatePicker;

public AuthForm(DatabaseHelper dbHelper)

{

InitializeComponent();

this.dbHelper = dbHelper;

InitializeUI();

}

private void InitializeUI()

{

Text = "Авторизация";

Size = new System.Drawing.Size(500, 400);

CenterToScreen();

var titleLabel = new Label

{

Text = "АВТОРИЗАЦИЯ",

Font = new System.Drawing.Font("Arial", 20, System.Drawing.FontStyle.Bold),

AutoSize = true,

Location = new System.Drawing.Point(150, 30)

};

Controls.Add(titleLabel);

var fullNameLabel = new Label { Text = "ФИО:", Location = new System.Drawing.Point(50, 100), AutoSize = true };

fullNameTextBox = new TextBox { Location = new System.Drawing.Point(150, 100), Width = 250 };

Controls.Add(fullNameLabel);

Controls.Add(fullNameTextBox);

var phoneLabel = new Label { Text = "Телефон:", Location = new System.Drawing.Point(50, 140), AutoSize = true };

phoneTextBox = new TextBox { Location = new System.Drawing.Point(150, 140), Width = 250 };

phoneTextBox.TextChanged += (s, e) => FormatPhoneNumber();

Controls.Add(phoneLabel);

Controls.Add(phoneTextBox);

var genderLabel = new Label { Text = "Пол:", Location = new System.Drawing.Point(50, 180), AutoSize = true };

genderComboBox = new ComboBox { Location = new System.Drawing.Point(150, 180), Width = 250 };

genderComboBox.Items.AddRange(new[] { "Мужской", "Женский" });

Controls.Add(genderLabel);

Controls.Add(genderComboBox);

var birthDateLabel = new Label { Text = "Дата рождения:", Location = new System.Drawing.Point(50, 220), AutoSize = true };

birthDatePicker = new DateTimePicker

{

Location = new System.Drawing.Point(150, 220),

Width = 250,

MaxDate = DateTime.Today.AddYears(-14),

MinDate = DateTime.Today.AddYears(-100)

};

Controls.Add(birthDateLabel);

Controls.Add(birthDatePicker);

var registerButton = new Button { Text = "Регистрация", Location = new System.Drawing.Point(150, 270), Width = 100 };

var loginButton = new Button { Text = "Вход", Location = new System.Drawing.Point(260, 270), Width = 100 };

Controls.Add(registerButton);

Controls.Add(loginButton);

var switchModeButton = new Button { Text = "У меня есть аккаунт", Location = new System.Drawing.Point(150, 310), Width = 210 };

Controls.Add(switchModeButton);

registerButton.Click += (s, ev) =>

{

if (!ValidateInputs())

return;

try

{

Console.WriteLine($"Попытка регистрации пользователя:");

Console.WriteLine($"ФИО: {fullNameTextBox.Text}");

Console.WriteLine($"Телефон: {phoneTextBox.Text}");

Console.WriteLine($"Пол: {genderComboBox.SelectedItem}");

Console.WriteLine($"Дата рождения: {birthDatePicker.Value.ToShortDateString()}");

var userId = dbHelper.RegisterUser(

fullNameTextBox.Text,

phoneTextBox.Text,

genderComboBox.SelectedItem.ToString(),

birthDatePicker.Value);

MessageBox.Show("Регистрация успешна!");

OpenMainForm(userId);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка регистрации: {ex.Message}");

}

};

loginButton.Click += (s, ev) =>

{

if (isLoginMode)

{

if (!ValidatePhoneNumber(phoneTextBox.Text))

{

MessageBox.Show("Введите корректный российский номер телефона");

return;

}

var userId = dbHelper.AuthenticateUser(phoneTextBox.Text);

if (userId.HasValue)

{

OpenMainForm(userId.Value);

}

else

{

MessageBox.Show("Пользователь не найден");

}

}

else

{

MessageBox.Show("Нажмите 'У меня есть аккаунт' для входа");

}

};

switchModeButton.Click += (s, ev) =>

{

isLoginMode = !isLoginMode;

fullNameLabel.Visible = !isLoginMode;

fullNameTextBox.Visible = !isLoginMode;

genderLabel.Visible = !isLoginMode;

genderComboBox.Visible = !isLoginMode;

birthDateLabel.Visible = !isLoginMode;

birthDatePicker.Visible = !isLoginMode;

registerButton.Visible = !isLoginMode;

switchModeButton.Text = isLoginMode ? "У меня нет аккаунта" : "У меня есть аккаунт";

titleLabel.Text = isLoginMode ? "ВХОД" : "АВТОРИЗАЦИЯ";

};

}

private bool ValidateInputs()

{

// Проверка ФИО

if (string.IsNullOrWhiteSpace(fullNameTextBox.Text) || !IsValidFullName(fullNameTextBox.Text))

{

MessageBox.Show("Введите корректное ФИО (только буквы и пробелы, минимум 2 слова)");

return false;

}

// Проверка телефона

if (!ValidatePhoneNumber(phoneTextBox.Text))

{

MessageBox.Show("Введите корректный российский номер телефона (+7XXXXXXXXXX или 8XXXXXXXXXX)");

return false;

}

// Проверка пола

if (genderComboBox.SelectedItem == null)

{

MessageBox.Show("Выберите пол");

return false;

}

// Проверка даты рождения

var age = DateTime.Today.Year - birthDatePicker.Value.Year;

if (birthDatePicker.Value.Date > DateTime.Today.AddYears(-age)) age--;

if (age < 14)

{

MessageBox.Show("Пользователь должен быть старше 14 лет");

return false;

}

else if (age > 100)

{

MessageBox.Show("Пользователь должен быть младше 100 лет");

return false;

}

return true;

}

private bool IsValidFullName(string fullName)

{

// Проверка что ФИО содержит только буквы и пробелы, и состоит минимум из 2 слов

return Regex.IsMatch(fullName, @"^[а-яА-ЯёЁa-zA-Z\s]{2,}(?:\s+[а-яА-ЯёЁa-zA-Z]{2,})+$");

}

private bool ValidatePhoneNumber(string phone)

{

// Проверка российского номера телефона (+7XXXXXXXXXX или 8XXXXXXXXXX)

return Regex.IsMatch(phone, @"^(\+7|8)[0-9]{10}$");

}

private void FormatPhoneNumber()

{

// Удаляем все нецифровые символы

string digitsOnly = Regex.Replace(phoneTextBox.Text, @"[^\d]", "");

// Ограничиваем длину

if (digitsOnly.Length > 11)

digitsOnly = digitsOnly.Substring(0, 11);

// Форматируем номер

if (digitsOnly.Length >= 1)

{

if (digitsOnly[0] == '7' || digitsOnly[0] == '8')

{

phoneTextBox.Text = $"+7{digitsOnly.Substring(1)}";

}

else

{

phoneTextBox.Text = $"+7{digitsOnly}";

}

}

// Устанавливаем курсор в конец

phoneTextBox.SelectionStart = phoneTextBox.Text.Length;

}

private void OpenMainForm(int userId)

{

var mainForm = new MainForm(dbHelper, userId);

mainForm.Show();

Hide();

}

private void InitializeComponent()

{

this.SuspendLayout();

//

// AuthForm

//

this.ClientSize = new System.Drawing.Size(482, 453);

this.Name = "AuthForm";

this.ResumeLayout(false);

}

}

}

**Приложение 3 (отчет о тестировании)**

**Приложение 4 (техническое задание)**

**1. Введение**

Настоящее техническое задание (ТЗ) описывает требования к разработке информационной системы электронной очереди для поликлиники. Система предназначена для автоматизации записи пациентов, управления очередями и отправки уведомлений, что позволит сократить время ожидания, минимизировать ошибки ручного учета и повысить качество обслуживания.

**2. Цели и задачи системы**

* Онлайн-запись пациентов: возможность записи через веб-приложение, выбор врача, даты и времени приема.
* Отслеживание очереди в реальном времени: отображение текущего статуса очереди
* SMS-оповещения: подтверждение записи, оповещение о вызове к врачу.
* Администрирование: управление расписанием врачей.

**3. Пользователи системы**

* Пациенты: запись на прием, просмотр статуса очереди
* Администраторы: настройка расписания врачей, управление очередями
* Врачи: просмотр списка пациентов
* Технические специалисты: мониторинг работы системы.

**4. Сценарии работы**

* Запись на прием: пациент выбирает врача и время, получает талон и SMS-подтверждение.
* Отслеживание очереди: пациент видит текущий статус на экране или в мобильном приложении.
* Формирование отчетов: администратор получает данные о загруженности врачей за период

**5. Интеграции**

* СМС-сервисы: интеграция с телефонами, SMS.ru или другими провайдерами.
* Электронная регистратура: обмен данными с существующей системой учета пациентов (при наличии).

**6. Требования к производительности и масштабируемости**

* Производительность: обработка до 100 записей в минуту, поддержка до 500 одновременных пользователей.
* Масштабируемость: возможность расширения для работы с несколькими филиалами поликлиники.

**7. Отказоустойчивость и безопасность**

* Доступность: 99.9% uptime.
* Безопасность: аутентификация через OAuth 2.0 / JWT, шифрование персональных данных (соответствие ФЗ-152).

Резервное копирование: ежедневное копирование данных.

**8. Визуализация данных**

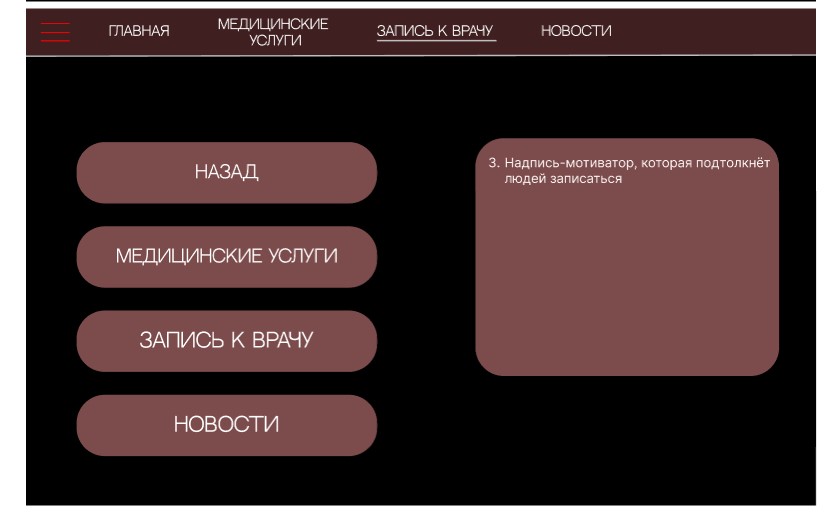
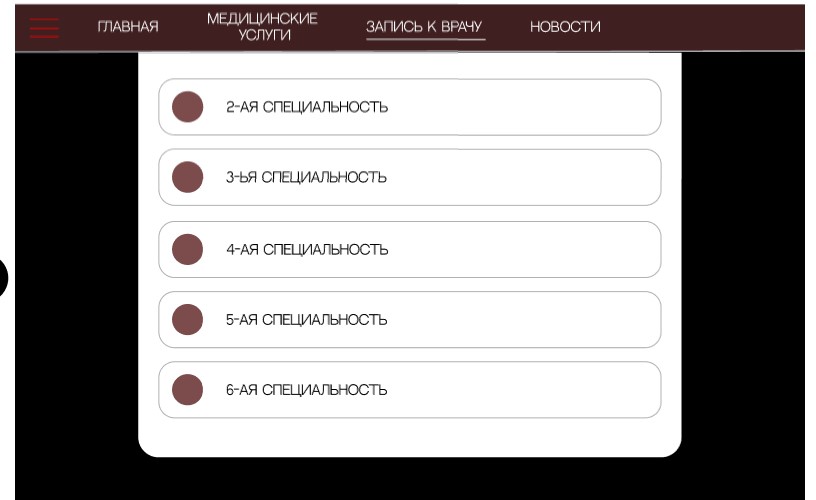
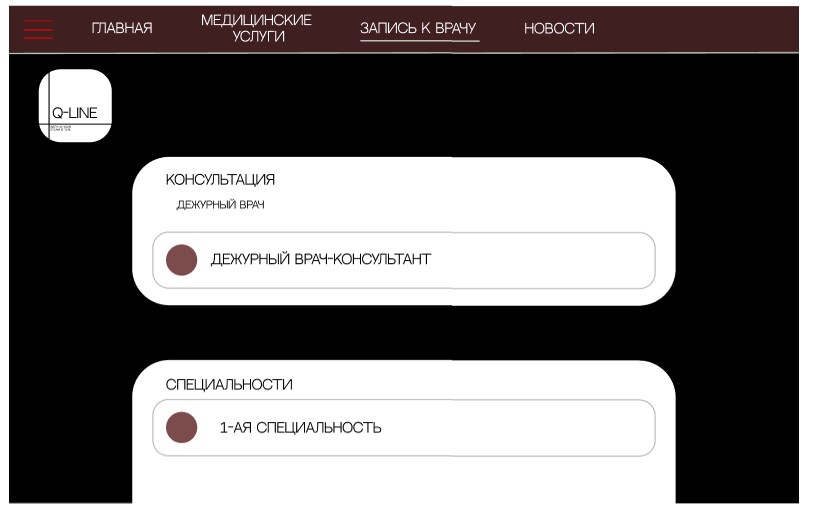
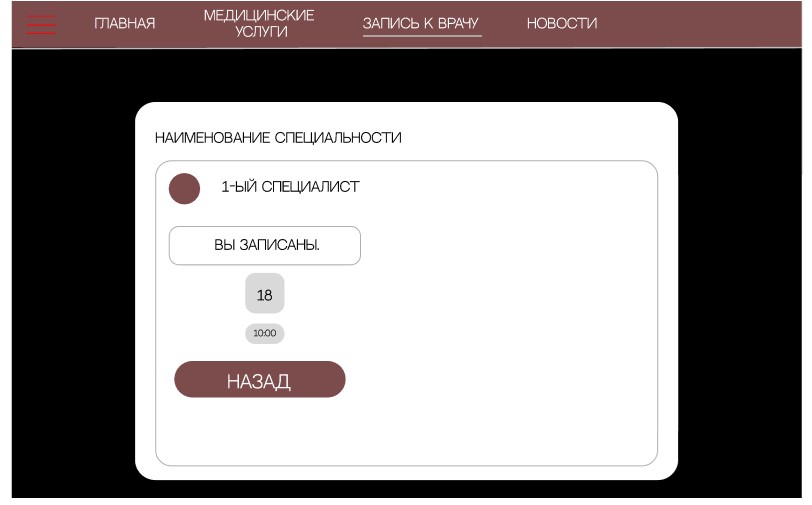
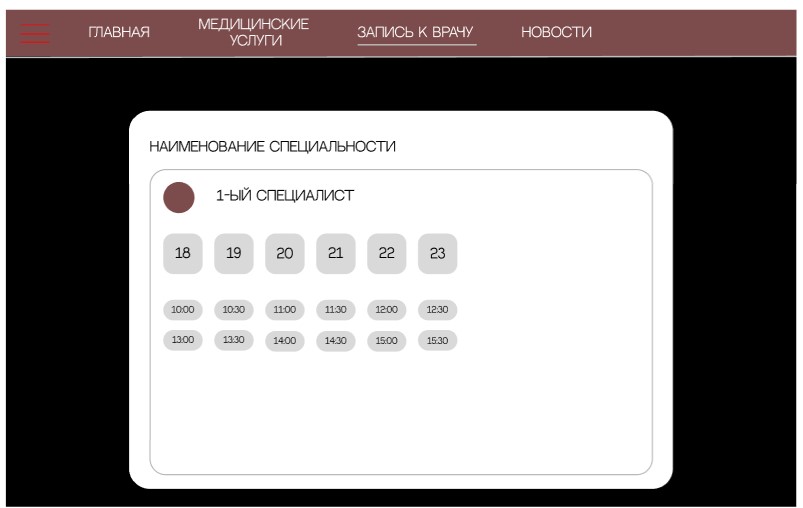
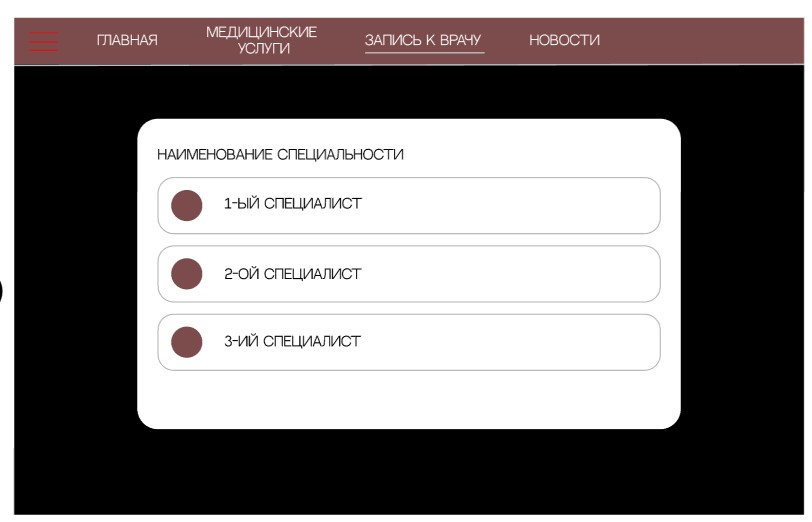
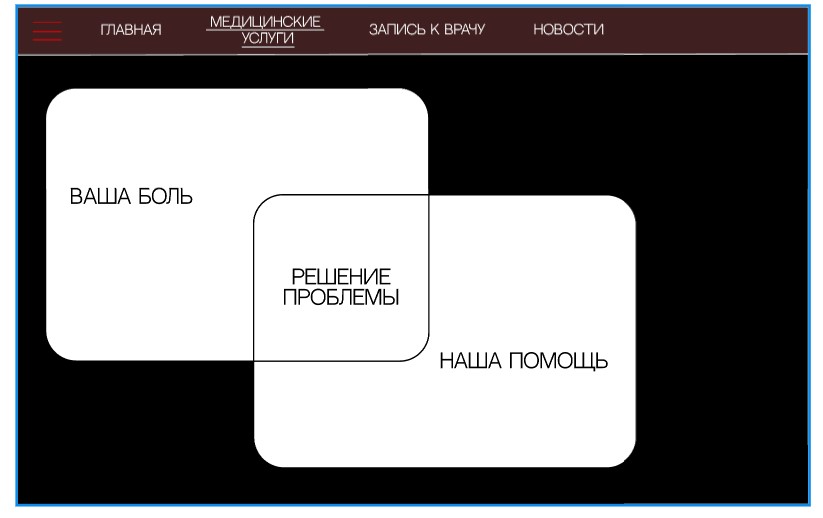
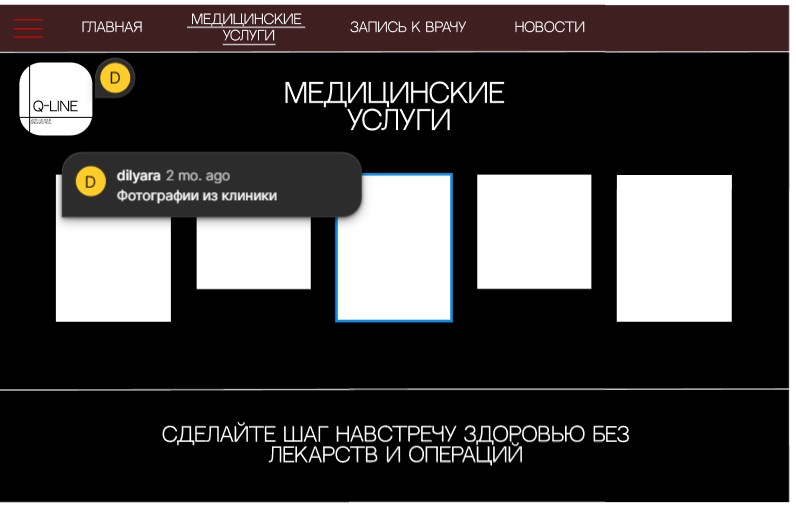
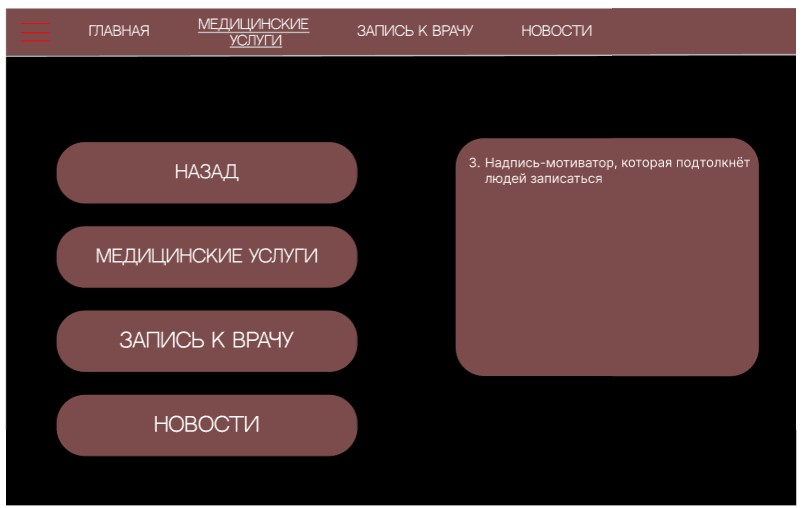
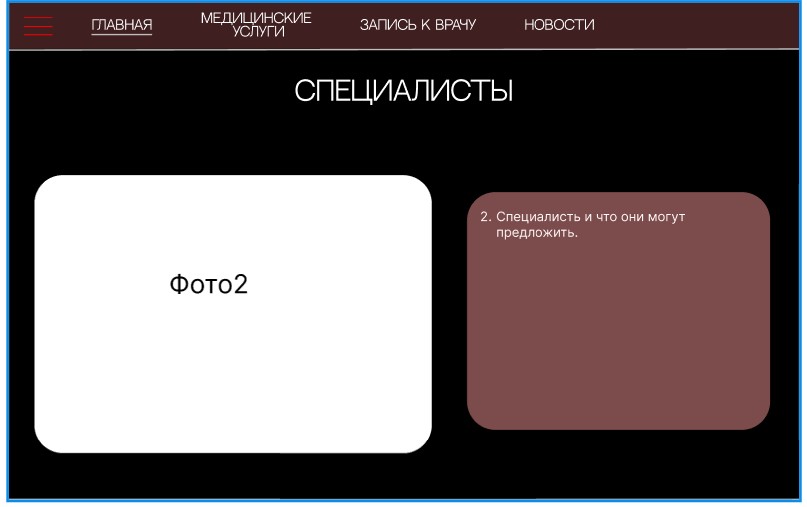
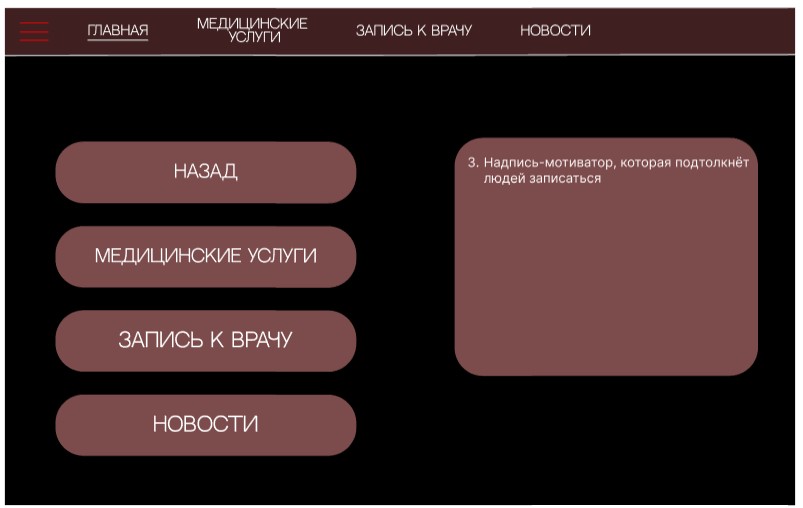
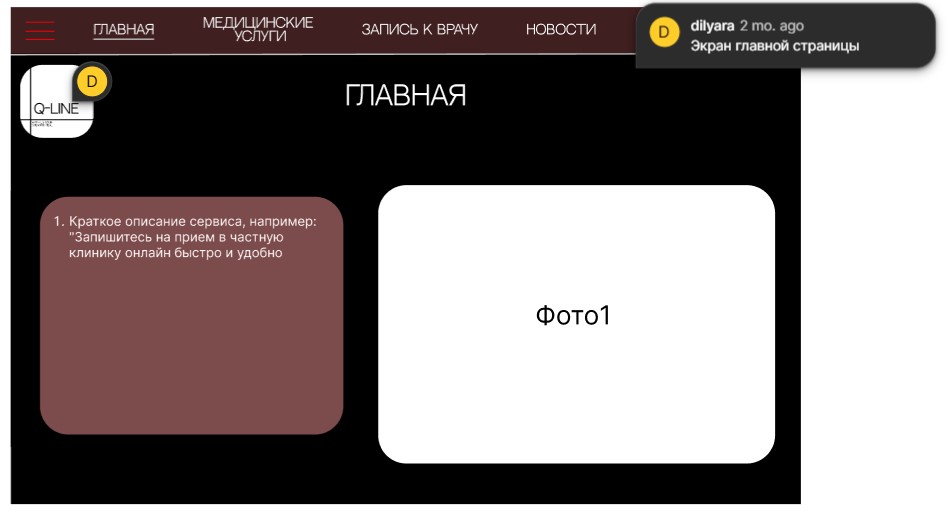
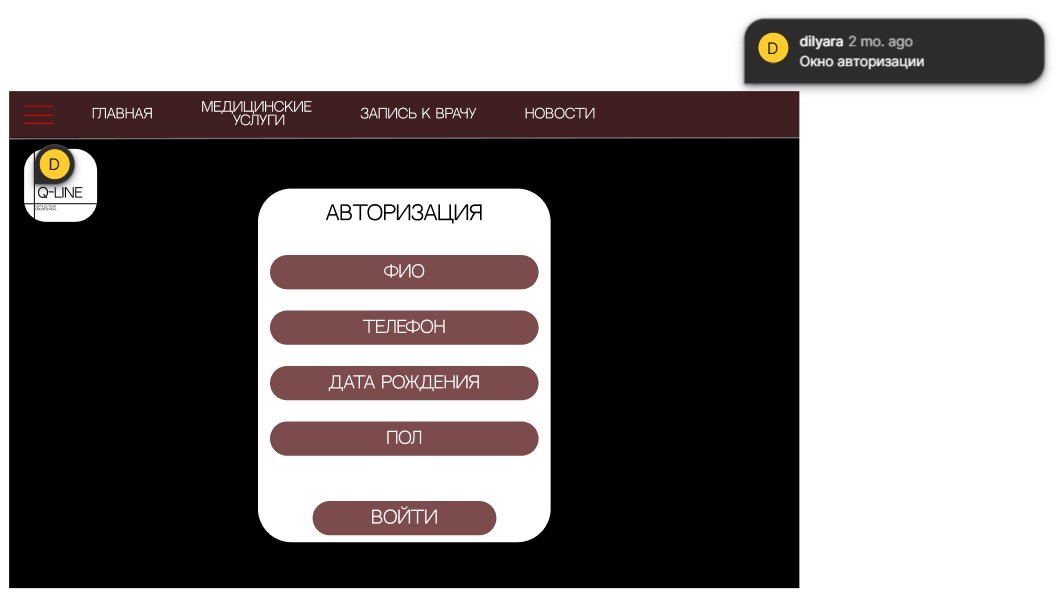
* Панель администратора: графики загруженности врачей, таблицы с историей записей.
* Экспорт отчетов: в форматы PDF, Excel.

**9. Технические требования**

* Серверная часть: язык: C# (ASP.NET Core)
* БД: PostgreSQL / SQL Server.
* SMS-шлюз: поддержка API (email, phone).

**10. Дизайн и интерфейс**

* Интуитивность: простое меню для пациентов (запись, отслеживание очереди), расширенные настройки для администраторов.
* Доступность: поддержка крупного шрифта и высокой контрастности.



**11. Тестирование**

Виды тестирования:

* Функциональное (проверка записи, уведомлений).
* Нагрузочное (обработка пиковых нагрузок).
* Интеграционное (проверка работы SMS-шлюза).

Критерии приемки: система соответствует всем требованиям ТЗ, отсутствие критических ошибок.

**12. Документация**

* Для пользователей: руководство по записи на прием (веб, мобильное приложение), инструкция для администраторов.
* Для разработчиков: API-документация.
* Описание архитектуры.

**13. Обучение и поддержка**

* Обучение: видеоинструкции для пациентов, вебинары для администраторов.
* Поддержка: чат, email, телефон.
* Время реакции: до 24 часов.

1. [organizaciya\_telefonnoy\_elektronnoy\_ocheredi\_ambulatornyh\_pacientov\_92\_v2](https://www.speechpro.ru/upload/solution/pdf/organizaciya_telefonnoy_elektronnoy_ocheredi_ambulatornyh_pacientov_92_v2.pdf?ysclid=manqqz0p6786373160)

   об электронной очереди [↑](#footnote-ref-1)
2. <https://forms.yandex.ru/u/6825edbc6d2d73048c75437f>

   Опрос для исследования проекта [↑](#footnote-ref-2)
3. <http://blog.foolsoft.ru/downloads/SmsRu.dll>

   Ссылка для скачивания библиотеки “SMS.ru” [↑](#footnote-ref-3)