

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
ГЛАВА 1. ТЕХНИЧЕСКИЙ ПРОЕКТ	7
1.1. Постановка задачи.....	7
1.2. Предпроектное исследование предметной области.....	7
1.3. Сравнительный анализ существующих приложений	9
1.4. Обоснования выбора инструментальных средств разработки.....	9
1.6. Требования к программному средству.....	14
1.6.1 Требования к функциональным характеристикам.....	15
1.6.2 Требования к структуре ПС	16
1.6.3 Требования к надежности.....	17
1.6.4 Условия эксплуатации	18
1.6.5 Системные требования к параметрам технических средств и информационной и программной совместимости	19
1.6.6 Требования к транспортировке и хранению.....	20
1.7. Требования к программной документации.....	21
1.8. Стадии и этапы разработки	21
ГЛАВА 2. РАБОЧИЙ ПРОЕКТ	24
2.1. Проектирование и реализация базы данных приложения.....	24
2.2. Разработка алгоритма решения задачи	24
2.2.1 Блок схема алгоритма	25
2.2.2 Диаграмма форм.....	26
2.2.3 Диаграмма классов.....	27
2.2.4 Диаграмма прецедентов.....	28
2.3. Описание разработки программного продукта	29
2.4. Разработка интерфейса программы	30
2.4.1 Интерфейс приложения	30
2.4.2 Интеграция с другими сервисами.....	31
2.4.3 Структурная схема приложения	32
2.5. Описание структуры входной и выходной информации	33
2.6. Отладка и тестирование приложения.....	34
2.6.1 Отладка и тестирование разработанного приложения и сайта.....	34

2.7. Публикация приложения на хостинге	38
2.8. Разработка руководства по использованию приложения.....	40
2.8.1 Введение по руководству по использованию приложения.....	42
2.8.2 Установка и описание основных действий для работы.....	42
2.8.3 Заключение по руководству по использованию приложения	43
ЗАКЛЮЧЕНИЕ.....	45
ЛИТЕРАТУРА	47
ПРИЛОЖЕНИЕ №1 «ЛИСТИНГ ПРОГРАММЫ».....	51

ВВЕДЕНИЕ

Актуальность разработки приложения для записи на прием в государственное казенное учреждение "Краевой центр социальной защиты населения" является необходимой мерой для улучшения качества обслуживания граждан и повышения эффективности работы учреждения.

Целями разработки приложения для записи на прием в государственное казенное учреждение "Краевой Центр социальной защиты населения" являются такие пункты как:

Во-первых, разрабатываемое приложение позволит сократить время ожидания в очереди на прием, что особенно важно для людей с ограниченными возможностями здоровья, пенсионеров и других категорий граждан, которые нуждаются в социальной поддержке.

Во-вторых, приложение будет предоставлять актуальную информацию о расписании работы центра, что позволит гражданам планировать свой визит заранее и не тратить время на ожидание в очереди.

В-третьих, приложение позволит ускорить процесс записи на прием, так как граждане смогут выбрать удобное время и дату посещения центра.

Таким образом, разработка приложения для записи на прием в государственный казенный центр социальной защиты населения позволит улучшить качество обслуживания граждан, сократить время ожидания и ускорить процесс записи на прием.

Задачи, которые должны быть решены в ходе разработки программы:

1. Провести анализ предметной области;
2. Рассмотреть аналоги создаваемой программы и их характеристики;
3. Выбрать средства разработки программного обеспечения;
4. Разработать информационно-логическую модель приложения;
5. Разработать требования к программному средству;
6. Разработать пользовательский интерфейс;
7. Спроектировать и создать базу данных;

8. Протестировать на работу программы;
9. Разработать руководство по использованию программы.

Объект исследования: государственное казенное учреждение "Краевой центр социальной защиты населения".

Предмет исследования: государственное казенное учреждение "Краевой центр социальной защиты населения" Забайкальского края города Краснокаменск.

Теоретическая и практическая значимость работы:

Теоретическая значимость заключается в том, что разработка приложения позволит упростить процесс записи на прием, ускорить обслуживание, сократить время ожидания в очереди и повысить качество обслуживания. Это также позволит снизить нагрузку на персонал и освободить время для более важных задач.

Практическая значимость заключается в том, что приложение может быть использовано не только в государственном казенном учреждении, но и в других организациях, предоставляющих услуги населению. Это позволит улучшить качество обслуживания и повысить уровень удовлетворенности клиентов.

Структура работы состоит из введения, основной части, заключения, списка литературы и приложений к данной работе.

Введение данной работы состоит из актуальности, целей, задач работы, объекта исследования, предмета исследования, теоретической и практической значимости работы, и структуры работы.

В список литературы записываются учебные материалы, а именно книги и учебники и Интернет-ресурсы, которые использовались при разработке программы и документации.

Приложения включают в себя вспомогательную документацию к разрабатываемому приложению.

ГЛАВА 1. ТЕХНИЧЕСКИЙ ПРОЕКТ

1.1. Постановка задачи

Согласно теме дипломной работы «Разработка информационной системы записи на приём в ГКУ КЦСЗН» целью работы является создание программы для записи посетителей на приём в государственное казенное учреждение «Краевой центр социальной защиты населения». Для разработки программного обеспечения потребуются две программы для программирования Visual Studio, Visual Studio Code и система управления базами данных PostgreSQL. В программе Visual Studio создаём программу, а в Visual Studio Code сайт. В системе управления базами данных PostgreSQL создаём таблицы информация, с которых будет перенесена в создаваемое приложение. Доступ к информации должен осуществляться без авторизации и регистрации так как пользователями данной программы будут сотрудники государственного казенного учреждения «Краевой центр социальной защиты населения» для которых нужен быстрый доступ к информации, находящейся в создаваемом приложении. Пользователь получивший доступ к информации, иметь возможность редактировать таблицу с информацией, а именно добавлять, изменять и удалять.

1.2. Предпроектное исследование предметной области

Для предпроектного исследования предметной области необходимо провести анализ текущей ситуации и определить требования к системе записи граждан на прием в государственное казенное учреждение "Краевой центр социальной защиты населения".

1. Анализ текущей ситуации.

Для начала необходимо определить, какие услуги предоставляет "Краевой центр социальной защиты населения" и как граждане записываются на прием за этими услугами.

2. Определение требований к системе.

На основе анализа текущей ситуации необходимо определить требования

к новой системе записи на прием граждан в "Краевой центр социальной защиты населения":

Удобство использования: система должна быть простой и интуитивно понятной в использовании.

Скорость обработки заявок: система должна обрабатывать заявки быстро и эффективно, чтобы минимизировать время ожидания для граждан.

Безопасность данных: система должна обеспечивать защиту персональных данных граждан и быть безопасной в использовании.

Интеграция с другими системами: система должна интегрироваться с другими системами, чтобы обеспечить полную функциональность и удобство использования.

3. Разработка требований к пользовательскому интерфейсу.

Необходимо разработать требования к пользовательскому интерфейсу системы записи на прием, который будет использоваться сотрудниками "Краевого центра социальной защиты населения". Интерфейс должен быть простым и интуитивно понятным, с возможностью выбора услуг, дат и времени приема.

4. Оценка рисков и возможностей.

После определения требований к системе необходимо оценить риски и возможности, связанные с ее внедрением. Это может включать оценку затрат времени на разработку, тестирование и внедрение системы, а также оценку возможных проблем, связанных с интеграцией с другими системами и безопасностью данных.

5. Разработка технического задания.

На основе требований к системе и оценки рисков и возможностей необходимо разработать техническое задание, которое будет использоваться для разработки и тестирования системы. Техническое задание должно содержать подробное описание всех требований к системе, а также описание процесса тестирования и приемки.

1.3. Сравнительный анализ существующих приложений

Существует несколько приложений для записи на прием в государственное казенное учреждение "Краевой центр социальной защиты населения". Рассмотрим их сравнительный анализ.

1. "Госуслуги" – это федеральное приложение, которое предоставляет возможность записи на прием к государственным учреждениям. Оно доступно в любом браузере на компьютере и для пользователей мобильных устройств, работающих на операционных системах iOS и Android;

2. "Электронный гражданин" – это приложение, разработанное Министерством цифрового развития, связи и массовых коммуникаций Российской Федерации. Оно позволяет записаться на прием в различные государственные учреждения, включая "Краевой центр социальной защиты населения";

3. "Мои документы" – это мобильное приложение, разработанное Федеральной налоговой службой Российской Федерации. С помощью него можно записаться на прием к налоговым органам, а также в другие государственные учреждения;

4. "МФЦ" – это многофункциональный центр, который предоставляет услуги по оформлению документов, выдаче справок и т. д. В приложении можно записаться на прием, выбрав удобное время и место.

1.4. Обоснования выбора инструментальных средств разработки

Для разработки потребуются две программы для программирования Visual Studio, Visual Studio Code и система управления базами данных PostgreSQL. В программе Visual Studio создаём программу, а в Visual Studio Code сайт. В система управления базами данных PostgreSQL создаём таблицы с информацией, далее хранящиеся данные будут перенесены в создаваемое приложение и сайт.

Visual Studio – это интегрированная среда разработки (IDE) для создания,

отладки и развертывания приложений на различных платформах, включая Windows, macOS и Linux. Visual Studio предлагает широкий набор инструментов и функций для разработчиков, которые позволяют им создавать и поддерживать высококачественные приложения.

Ниже представлены некоторые из основных причин, почему Visual Studio выбран для разработки создаваемого мною приложения:

1. Широкий спектр возможностей: Visual Studio предоставляет широкий спектр инструментов и функций, которые охватывают все этапы разработки приложений. Это включает в себя поддержку языков программирования, таких как C++, C#, Java и другие, а также инструменты для тестирования, отладки, сборки и развертывания;

2. Интеграция с другими инструментами: Visual Studio интегрируется с другими инструментами разработки, такими как Git, Team Foundation Server и Azure DevOps Services, что упрощает управление проектами и совместную работу над ними;

3. Высокая производительность: Visual Studio использует современные технологии для обеспечения высокой производительности и скорости разработки. Она имеет встроенный отладчик, который позволяет быстро находить и исправлять ошибки в коде, а также поддерживает многопоточную отладку, что ускоряет процесс отладки;

4. Поддержка сообщества: Visual Studio имеет активное сообщество пользователей и разработчиков, которое предоставляет множество ресурсов и информации для обучения, обмена опытом и решения проблем. Существует множество форумов, видеоуроков и онлайн-курсов, которые помогают разработчикам освоить Visual Studio и использовать ее на полную мощность.

Visual Studio Code (VS Code) – это интегрированная среда разработки (IDE) с открытым исходным кодом, разработанная Microsoft. Она является одним из наиболее популярных инструментов для разработки программного обеспечения. Вот несколько причин, почему выбор VS Code может быть обоснован:

1. Кроссплатформенность: VS Code поддерживает множество операционных систем, включая Windows, macOS и Linux. Это позволяет разработчикам использовать один и тот же инструмент на разных платформах;

2. Гибкость: VS Code предоставляет множество возможностей для настройки и расширения функциональности. Разработчики могут добавлять плагины и расширения, чтобы адаптировать среду под свои потребности;

3. Поддержка языков программирования: VS Code имеет встроенную поддержку многих языков программирования, включая C++, JavaScript, TypeScript, Python, Java, Go, и многие другие. Это делает его универсальным инструментом для разработки программного обеспечения на различных языках;

4. Интеграция с Git: VS Code интегрируется с системой контроля версий Git, что позволяет разработчикам легко управлять своими проектами и отслеживать изменения;

5. Простота использования: VS Code прост в использовании и не требует специальных знаний или навыков. Он предоставляет удобный интерфейс и множество подсказок, которые помогают разработчикам быстро начать работу;

6. Доступность: VS Code бесплатен и доступен для загрузки на различных платформах. Это делает его доступным для широкого круга разработчиков.

В целом, VS Code является мощным и гибким инструментом для разработки программного обеспечения, который может быть использован как начинающими, так и опытными разработчиками.

Для разработки PostgreSQL можно использовать различные инструментальные средства, такие как IDE (Integrated Development Environment), текстовые редакторы, системы контроля версий и т. д. Ниже приведены некоторые обоснования для выбора того или иного инструментального средства:

1. IDE (Integrated Development Environment) предоставляют полный набор инструментов для разработки баз данных, включая редактор кода, отладчик, инструменты для создания и управления базами данных и т. д. Они также предоставляют доступ к документации и справочным материалам, что упрощает

процесс обучения и разработки. IDE обычно являются платными инструментами, но они могут быть очень полезными для разработчиков, особенно если они работают над большими проектами;

2. Текстовые редакторы: Текстовые редакторы также могут использоваться для разработки PostgreSQL. Они предоставляют базовые возможности для написания кода и редактирования файлов. Некоторые текстовые редакторы имеют встроенный отладчик и инструменты для управления базами данных. Однако они не предоставляют все необходимые инструменты для разработки баз данных и могут быть менее удобными, чем IDE;

3. Системы контроля версий: Системы контроля версий позволяют разработчикам работать над проектом совместно и управлять изменениями кода. Для PostgreSQL можно использовать системы контроля версий, такие как Git. Они позволяют отслеживать изменения в коде, создавать резервные копии и восстанавливать проект в случае необходимости;

4. Инструменты командной строки: для разработки PostgreSQL также можно использовать инструменты командной строки. Они предоставляют простой интерфейс для выполнения команд и работы с базами данных. Например, можно использовать утилиты `psql` и `pgAdmin` для работы с PostgreSQL. Однако инструменты командной строки могут быть менее удобны, чем IDE или текстовые редакторы.

1.5. Разработка информационно-логической модели приложений

Информационно-логическая модель приложений для записи на прием в государственное казенное учреждение "Краевой центр социальной защиты населения" должна учитывать следующие аспекты:

1. Запись на прием: приложение должно предоставлять гражданам возможность выбрать дату и время приема, а также указать, на какую услугу они хотят записаться;

2. Отмена записи: в случае необходимости, гражданин может отменить

свою запись на прием;

3. Интеграция с другими системами: приложение должно быть интегрировано с другими системами, такими как система управления услугами, чтобы обеспечить более эффективную работу учреждения и удобство для граждан.

Логическая модель данных – это представление данных, которое не зависит от конкретной реализации системы управления базами данных (СУБД). Она определяет структуру данных, их связи и ограничения, а также операции, которые можно выполнять над ними. Логическая модель описывает данные таким образом, чтобы они были независимыми от конкретной СУБД и могли быть использованы в различных системах.

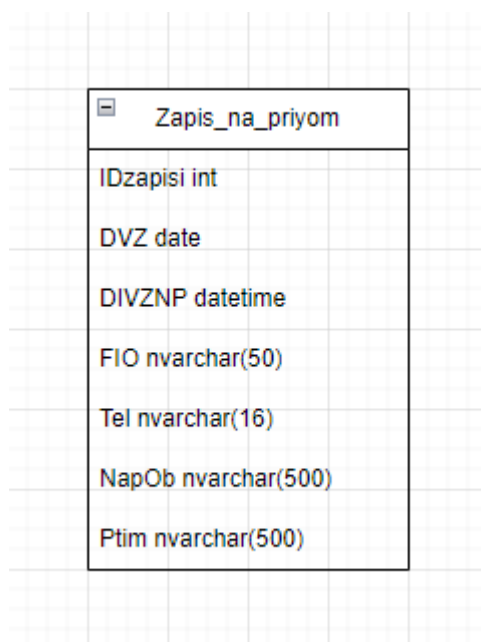


Рисунок 1 – Логическая модель данных

Физическая модель данных – это отображение логической модели данных в конкретную реляционную схему базы данных. Физическая модель содержит информацию о том, как данные хранятся на диске и как организованы таблицы и индексы. Физическая модель зависит от конкретной СУБД, поскольку каждая СУБД имеет свои особенности и ограничения.

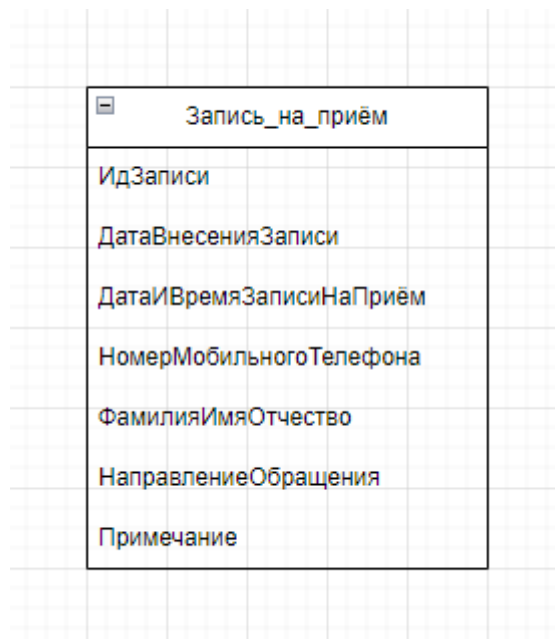


Рисунок 2 – Физическая модель данных

1.6. Требования к программному средству

Для того, чтобы разработать программное средство, необходимо определить его требования. Требования к программному средству могут быть функциональными и нефункциональными.

Функциональные требования описывают, какие функции должно выполнять программное средство для решения задач пользователя.

Нефункциональные требования описывают свойства программного средства, которые не связаны напрямую с его функциональностью. К ним могут относиться такие параметры, как производительность, надежность, безопасность, удобство использования и т. д.

Также следует учитывать требования к совместимости программного средства с другими системами и платформами, а также к его масштабируемости и готовности к изменениям в будущем.

Определение требований является важным этапом в разработке программного средства, поскольку правильно сформулированные требования обеспечивают более эффективную и успешную разработку и внедрение программного продукта.

1.6.1 Требования к функциональным характеристикам

Для разработки программного продукта необходимо определить требования к его функциональным характеристикам. Функциональные характеристики определяют, какие действия и задачи должен выполнять программный продукт.

Некоторые общие требования к функциональным характеристикам могут включать в себя:

1. **Функциональность.** Программа должна выполнять все задачи и действия, для которых она была разработана. При этом эти действия должны быть выполнены правильно и эффективно;

2. **Надежность.** Программа должна работать стабильно и надежно. Она должна обрабатывать данные правильно и предоставлять верные результаты в любых условиях;

3. **Эффективность.** Программа должна работать быстро и не занимать слишком много ресурсов системы. Она должна быть оптимизирована для работы на всех платформах и устройствах;

4. **Удобство использования.** Программа должна быть простой и удобной в использовании для конечного пользователя. Она должна иметь интуитивно понятный интерфейс и легко доступную документацию;

5. **Совместимость.** Программа должна быть совместима со всеми необходимыми платформами, операционными системами и другими программными продуктами;

6. **Безопасность.** Программа должна обеспечивать безопасность данных и пользователей. Она должна иметь средства защиты от взлома и сбоев;

7. **Масштабируемость.** Программа должна быть способна работать с большим объемом данных и увеличиваться в масштабе в соответствии с

потребностями организации.

Это только некоторые общие требования к функциональным характеристикам, их полный список может зависеть от конкретного проекта и его целей.

1.6.2 Требования к структуре ПС

Структура ПС (программного средства) должна отвечать требованиям эффективности, удобства использования и поддержки.

Вот несколько основных требований к структуре ПС:

1. Модульность: Система должна быть разбита на модули, которые могут быть разработаны и изменены независимо друг от друга. Это позволяет улучшить модифицируемость, гибкость и повторное использование компонентов;

2. Иерархия: Система должна быть организована с помощью иерархии модулей для обеспечения четкой и логической структуры. Иерархия также может помочь при определении зависимостей между модулями;

3. Целостность: Система должна обеспечивать целостность данных, которые могут пересекаться между модулями или храниться в различных базах данных. Это требование подразумевает соответствие данных, их сохранение и защиту от ошибок;

4. Взаимодействие: Система должна обеспечивать взаимодействие между модулями и пользователями. Например, системы должны иметь возможность взаимодействовать с другими системами или базами данных через стандартные протоколы, API и интерфейсы;

5. Устойчивость: Система должна быть устойчивой к ошибкам и неисправностям. Это может быть достигнуто за счет использования методов бэкапирования, планирования восстановления, а также тестирования на

прочность и отказоустойчивости;

6. Безопасность: Система должна быть безопасной и защищенной от взломов, атак или других угроз безопасности;

7. Поддержка: Система должна быть легко поддерживаемой и масштабируемой. Это требование подразумевает возможность быстрого обновления системы, решения проблем пользователей и поддержки работы системы на различных платформах и операционных системах.

1.6.3 Требования к надежности

Надёжность программных средств – это способность работать без сбоев и ошибок на протяжении длительного времени.

Основные требования к надежности программных средств:

1. Устойчивость к ошибкам: программа должна работать без ошибок и сбоев. В случае возникновения ошибок они должны быть быстро исправлены;

2. Устойчивость к вредоносным программам: программное обеспечение должно быть защищено от вредоносных программ, таких как вирусы, трояны, шпионские программы и т. д;

3. Совместимость: программа должна работать с другими программами и операционными системами без проблем;

4. Масштабируемость: программа должна быть способна обрабатывать большие объемы данных и работать с большим количеством пользователей;

5. Резервное копирование: дополнительное хранение данных и программного обеспечения для восстановления в случае возникновения неполадок;

6. Безопасность: проектирование программы с учетом защиты от несанкционированного доступа к программному обеспечению и данным, обеспечение конфиденциальности и целостности данных;

7. Документирование и обучение: создание документации, инструкций и учебных пособий, обеспечение знаний пользователей и разработчиков, чтобы обеспечить правильное использование программного обеспечения.

1.6.4 Условия эксплуатации

Условия эксплуатации компьютера могут варьироваться в зависимости от конкретных требований производителя и окружающей среды. Однако, в целом, для обеспечения оптимальной работы компьютера следует учитывать следующие факторы:

1. Температура: Компьютер должен работать при комнатной температуре (от 20°C до 24°C). Не рекомендуется эксплуатация компьютера в слишком жаркой или холодной комнате;

2. Вентиляция: Обеспечьте достаточное пространство вокруг компьютера для свободного потока воздуха и охлаждения системы. Не ставьте компьютер в узкие, закрытые пространства, такие как шкафы, чтобы не нарушить естественный поток воздуха;

3. Пыль и грязь: регулярно очищайте корпус компьютера от пыли и грязи. Это поможет избежать перегрева и поломок;

4. Влажность: Компьютер не должен работать в слишком сырой или влажной среде. При высокой влажности возможны коррозия и короткие замыкания;

5. Питание: для обеспечения стабильной работы компьютера используйте надежный и стабильный источник питания, который соответствует требованиям производителя;

6. Электростатический разряд: перед началом любых работ с компьютером рекомендуется разрядить статическое электричество из вашего тела. Заземление при проведении работ внутри системного блока также может быть полезным для предотвращения повреждения компонентов;

7. Хранение: Компьютер следует хранить в чистом и сухом месте, где он не подвергается воздействию воды, магнитных полей или других вредных факторов;

8. Условия эксплуатации могут также зависеть от конкретных требований производителя и подразумевать дополнительные рекомендации по обработке, управлению и защите компьютера.

1.6.5 Системные требования к параметрам технических средств и информационной и программной совместимости

Системные требования к параметрам технических средств и информационной и программной совместимости зависят от конкретного программного обеспечения.

Однако, обычно системные требования включают в себя следующие характеристики компьютера:

Процессор: как минимум двухъядерный процессор с тактовой частотой не менее 2 ГГц.

Оперативная память (RAM): как минимум 4 ГБ для большинства приложений, но для более сложных задач может потребоваться до 8 ГБ или более.

Жесткий диск: свободное место на жестком диске зависит от того, сколько данных вы планируете хранить. В целом, для установки операционной системы и приложений нужно как минимум 20 ГБ свободного пространства.

Видеокарта: большинство приложений поддерживают интегрированные видеокарты, но для выполнения графических задач с высокой производительностью может потребоваться дискретная видеокарта.

Монитор: разрешение экрана зависит от ваших потребностей. Рекомендуется выбирать монитор с разрешением Full HD (1920 x 1080) или выше.

Операционная система: необходимо убедиться, что ваша операционная система поддерживается программным обеспечением, которое вы планируете использовать.

Важно также убедиться в информационной и программной совместимости. Проверьте, что используемые вами приложения и операционная система поддерживаются друг другом и что они совместимы с версией железа, на которой вы работаете.

Для данной программы рекомендуются следующие параметры:

Таблица 1 — «Системные требования к ПК»

Компонент компьютера	Рекомендуемые требования к ПК
Операционная система:	Windows 10 (86 или 64 разрядная)
CPU:	Intel Core i3 3217U 1.8Ghz двух ядерный, с видео ядром Intel® HD Graphics 4000
Свободное место на жёстком диске:	1ГБ
Оперативная память:	2ГБ

1.6.6 Требования к транспортировке и хранению

Транспортировка должна осуществляться согласно международному стандарту ГОСТ 21552–84.

Для транспортировки ПК рекомендуется выполнить следующие действия:

1. Выключите компьютер, расположенный на вертикальной поверхности;
2. Отсоедините все провода и кабели, подключенные к системному блоку;
3. Убедитесь, что материнская плата, карты расширения, жесткие диски и прочие устройства правильно установлены и надежно закреплены;
4. Если возможно, удалите некоторые компоненты, такие как карты памяти, СД-приводы и другие, которые могут легко отстегиваться и упакуйте их отдельно;
5. При помощи специального короба или упаковочного материала, надежно зафиксируйте системный блок, чтобы он не мог двигаться внутри коробки;

6. Убедитесь, что коробка закрыта и узлы транспортировки зафиксированы;
7. Переверните коробку на бок и осторожно переносите ее в машину;
8. Перенесите коробку в место назначения, следуя указаниям по складированию;
9. При распаковке ПК тщательно проверьте все компоненты, характеристики работы и состояние системы в целом.

Хранение программного обеспечения осуществляется на специальных устройствах таких как:

1. жёсткий диск (HDD);
2. внешний жёсткий диск (EHD);
3. твердотельные накопители (SSD);
4. сетевой накопитель (NAS).

1.7. Требования к программной документации

Для данного программного продукта было разработано техническое задание, пояснительная записка и презентация. Техническое задание – это документ проекта, в котором указаны назначение, технические характеристики и стадии разработки проекта. Пояснительная записка – это изложение информации из плана проекта. Презентация – это документ, предназначенный для представления чего-либо в данном случае представление проделанной работы.

1.8. Стадии и этапы разработки

Важно помнить, что разработка приложения – это итеративный процесс, который может включать в себя повторение некоторых стадий для улучшения функциональности или исправления ошибок. Кроме того, необходимо учитывать особенности данной области, такие как защита персональных данных, чтобы

создаваемое приложение соответствовало всем требованиям и регуляциям.

Таблица 2 «Стадии и этапы разработки»

№ п/п	Стадии разработки	Этапы работы Содержание работы
1	Исследования и анализирование	Исследовать и проанализировать аналоги разрабатываемого программного обеспечения Постановка целей и задач. Сбор материалов исследования. Выбор критериев на основании проведённого анализа других программ и веб-сайтов. Обоснование необходимости разработки своего программного обеспечения
2	Проектирование ПО	Определение структуры входных и выходных данных. Обоснование целесообразности применения ранее разработанных программ. Определение требований к техническим средствам. Обоснование принципиальной возможности решения поставленной задачи. Предварительное создание структуры входных и выходных данных. Планирование разработки программного обеспечения путём выбора методов решения задач. Разработка программной документации общего описания алгоритма решения задач.
3	Разработка ПО	Определение требований к программе. Определение стадий, этапов и сроков разработки программы и документации на разрабатываемое программное обеспечение. Выбор языков программирования для разрабатываемого программного обеспечения. Определение необходимости проведения научно-исследовательских работ на последующих стадиях разработки. Согласование и утверждение технического задания. Разработка алгоритма решения задач. Создание структуры входных и выходных данных. Определение синтаксиса языка. Разработка структуры программы. Окончательное определение конфигурации технических средств.

4	Тестирование ПО	<p>Разработка программного обеспечения, согласование и утверждение программы и методики испытаний. Проведение предварительных государственных, межведомственных, приемо-сдаточных и других видов испытаний. Корректировка программы и программной документации по результатам испытаний.</p> <p>Предварительная проверка работоспособности программного обеспечения.</p>
5	Сопровождение и внедрение ПО	<p>Подготовка к сдаче программного обеспечения и последующая её передача.</p> <p>Подготовка к сдаче программного обеспечения и последующая её передача и программной документации для сопровождения и (или) изготовления. Оформление и утверждение акта о передаче программы на сопровождение и (или) изготовление. Передача программы в фонд алгоритмов и программ.</p>

ГЛАВА 2. РАБОЧИЙ ПРОЕКТ

2.1. Проектирование и реализация базы данных приложения

Для проектирования и реализации базы данных приложения для записи граждан на прием в государственное казенное учреждение "Краевой центр социальной защиты населения" необходимо выполнить следующие шаги:

1. Определить требования к базе данных. Необходимо определить, какие данные будут храниться в базе данных, какой объем данных будет храниться, как часто будут производиться обновления данных и т. д;
2. Создать структуру базы данных. Структура базы данных должна быть оптимизирована для хранения и поиска необходимой информации. Она должна содержать таблицы для хранения информации о дате внесения записи в базу данных, назначенного времени на приём, фамилия имя отчество гражданина, мобильный номер телефона гражданина, цель визита и номер кабинета, и примечание при наличии;
3. Разработать SQL запросы для работы с базой данных. SQL-запросы должны быть написаны на языке SQL и использоваться для создания, изменения и удаления данных в таблицах;
4. Создать хранимые процедуры и функции для обработки данных. Хранимые процедуры и функции могут использоваться для выполнения сложных операций, таких как поиск данных, обновление данных и т. д;
5. Протестировать приложение на различных устройствах. Приложение должно работать корректно на различных устройствах, таких как компьютеры.

2.2. Разработка алгоритма решения задачи

Для разработки алгоритма решения задачи нужно:

1. Понять постановку задачи и ее условия;
2. Выбрать подходящий метод решения задачи;
3. Разбить задачу на составные части или подзадачи, если это возможно;

4. Написать алгоритм для каждой из подзадач;
5. Объединить все подзадачи в единый алгоритм решения задачи.

Для написания алгоритма используются различные языки программирования, такие как C#, JavaScript и другие. Важно выбрать язык, который лучше всего подходит для решения данной задачи.

При написании алгоритма необходимо учитывать специфику данных, с которыми работает алгоритм, чтобы он был максимально эффективным и точным. Также необходимо учитывать ограничения по времени и памяти, которые могут быть наложены на решение задачи.

После написания алгоритма следует провести тестирование и отладку программы, чтобы убедиться в ее правильности и корректности работы на разных входных данных.

2.2.1 Блок-схема алгоритма

Блок-схема представляет собой совокупность символов, соответствующих этапам работы алгоритма и соединяющих их линий. Пунктирная линия используется для соединения символа с комментарием. Сплошная линия отражает зависимости по управлению между символами и может снабжаться стрелкой. Стрелку можно не указывать при направлении дуги слева направо и сверху вниз. Линии должны подходить к символу слева, либо сверху, а исходить снизу, либо справа.

Есть и другие типы линий, используемые, например, для изображения блок-схем параллельных алгоритмов они, как и ряд специфических символов, не рассматриваются.

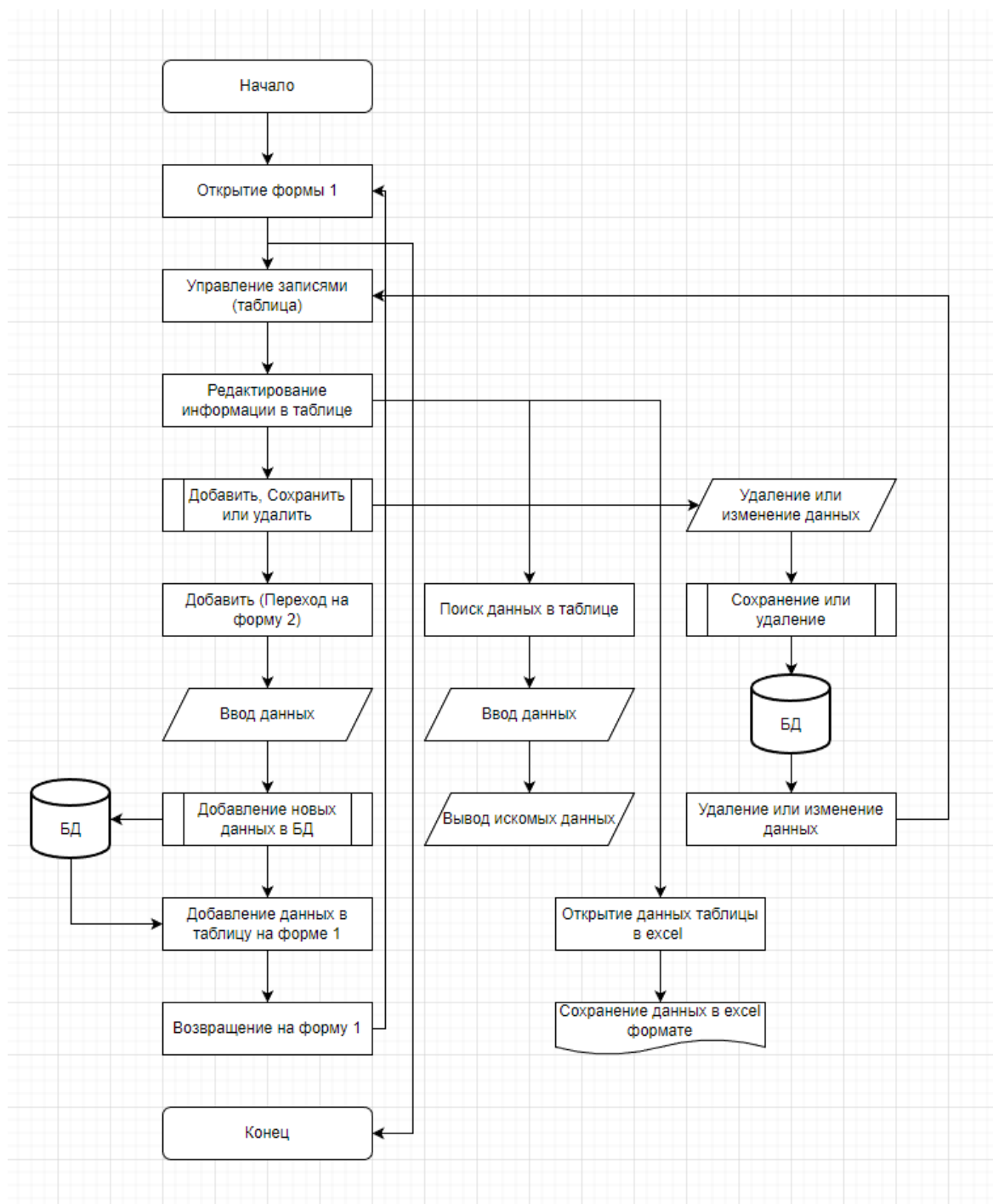


Рисунок 3 – Блок схема алгоритма

2.2.2 Диаграмма форм

Диаграмма форм – это графическое изображение компонентов и связей между ними в системе. Ее разработка необходима для того, чтобы представить функционирование системы более наглядно и понятно.

Данная диаграмма может содержать информацию о структуре системы, а

также о способах взаимодействия между компонентами. Для этого на диаграмме могут быть использованы различные символы и обозначения.

Другим важным аспектом разработки диаграммы форм является возможность выявления проблемных зон в системе и определение потенциальных рисков. Таким образом, диаграмма форм может быть полезным инструментом для улучшения работы и оптимизации функционирования системы.

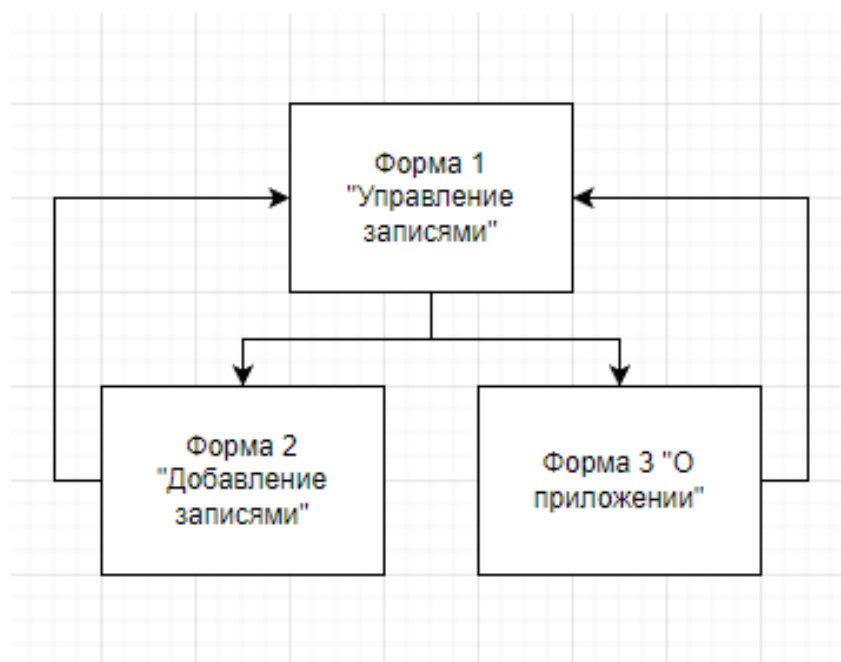


Рисунок 4 – Блок схема алгоритма

2.2.3 Диаграмма классов

Диаграмма классов – это графический инструмент, который используется в объектно-ориентированном программировании для описания классов и их отношений между собой. Она позволяет представить структуру программы в виде набора классов и интерфейсов, а также показать связи между ними.

На диаграмме классов каждый класс представляется в виде прямоугольника, в котором указывается название класса, его атрибуты (переменные) и методы (функции).

Диаграммы классов используются при проектировании программного обеспечения для лучшего понимания структуры программы и ее компонентов, а также для идентификации потенциальных проблем в разработке. Они могут быть использованы как на этапе проектирования программы, так и в ходе ее дальнейшей разработки и поддержки.

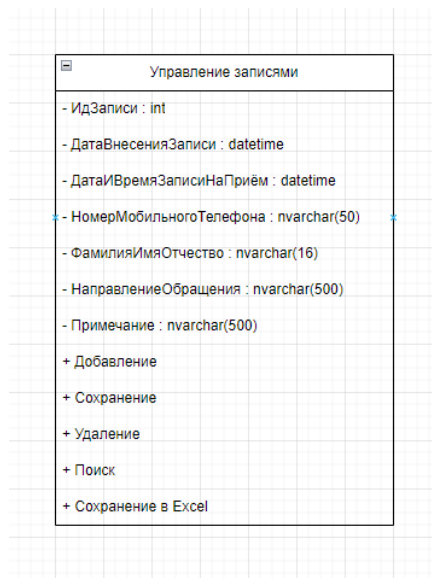


Рисунок 5 – Биagramма классов

2.2.4 Диаграмма прецедентов

Диаграмма прецедентов (Use Case Diagram) – это диаграмма, которая отображает отношения между актерами (пользователями системы) и сценариями использования (прецедентами), в которых эти актеры участвуют. Диаграмма прецедентов используется для моделирования функциональности системы и ее интерфейса. Она является частью Unified Modeling Language (UML), который широко используется при разработке программного обеспечения.

На диаграмме прецедентов каждый прецедент представлен в виде эллипса, а актеры - прямоугольниками или моделью человека. Связи между прецедентами и актерами обозначают типы взаимодействия пользователей с системой. Также на диаграмме можно указать расширения и включения прецедентов, что позволяет более подробно описать функциональность системы.

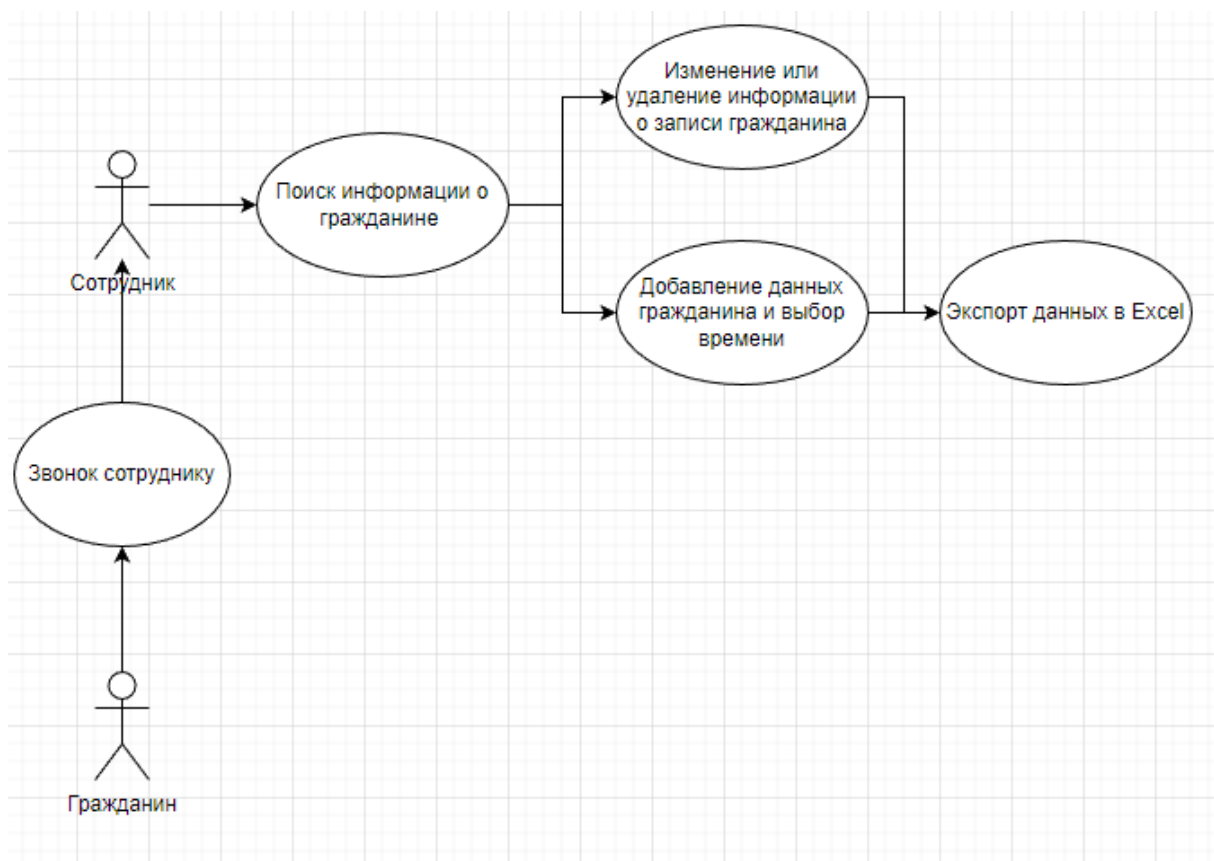


Рисунок 6 – Диаграмма прецедентов

2.3. Описание разработки программного продукта

Разработка программного продукта – это процесс создания программы, которая будет выполнять определенные задачи или функции.

Первым этапом разработки является анализ требований к программному продукту. На этом этапе определяются цели и задачи, которые должна выполнять программа и сайт, а также требования к ее функциональности, дизайну и производительности.

После анализа требований начинается разработка программного кода. Разработчики используют языки программирования и инструменты для создания кода, который будет выполнять необходимые функции.

Затем проходит тестирование, чтобы убедиться, что он работает правильно и соответствует требованиям.

Далее следует этап интеграции и тестирования, когда программа

объединяется с другими системами и приложениями, а также проверяется ее совместимость с уже существующими технологиями.

Наконец, после успешного тестирования и интеграции, продукт готов к использованию. Продукт может быть выпущен как самостоятельный продукт или как часть более крупного проекта.

Разработка программного обеспечения осуществляется на основании технического задания. Данное программное обеспечение предназначено для записи на прием в государственное казенное учреждение "Краевой центр социальной защиты населения". В программном плане приложение представляет собой окно с информацией в табличном виде с возможностью редактирования данной информации, а также вспомогательные инструменты такие как поиск и сортировка.

2.4. Разработка интерфейса программы

Общая структура приложений может выглядеть так:

1. Пользовательский интерфейс (UI) – это то, что пользователи видят и с чем взаимодействуют. UI состоит из элементов, таких как кнопки, поля ввода, метки, изображения и т. д.

2. Интеграция с другими системами (Integration) – это модуль, который позволяет приложению взаимодействовать с другими системами, такими как социальные сети, мессенджеры или сервисы для обработки платежей.

Это общая структура приложения, но конкретная структура может различаться в зависимости от конкретного приложения и использованных технологий.

2.4.1 Интерфейс приложения

Разработка интерфейса – это важный этап в создании программы. Интерфейс должен быть удобным, интуитивно понятным и соответствовать

потребностям пользователей.

При разработке интерфейса необходимо учитывать следующие моменты:

1. Используйте простые и понятные иконки и шрифты;
2. Разделите интерфейс на логические блоки, чтобы пользователи могли быстро найти нужную информацию;
3. Убедитесь, что интерфейс адаптивен;
4. Проведите тестирование интерфейса, чтобы убедиться в его удобстве и эффективности.

2.4.2 Интеграция с другими сервисами

Интеграция – это процесс объединения различных систем, сервисов или компонентов в единую систему, чтобы они могли работать вместе и обмениваться данными.

В контексте веб-разработки интеграция означает объединение различных сервисов и приложений на сайте или веб-приложении. Например, это может быть интеграция платежных систем для обработки онлайн-платежей, интеграция социальных сетей для авторизации пользователей или интеграция сторонних приложений для реализации дополнительной функциональности.

Интеграция может быть осуществлена как программно, путем написания кода, который связывает различные системы, так и с помощью использования стандартных протоколов и API (интерфейсов программирования приложений) для обмена данными между системами.

Эффективная интеграция позволяет значительно расширить возможности сайта или приложения и повысить его удобство для пользователей.

В данном случае приложение связано с сайтом общей базой данных. Если оставить заявку на сайте, то она отобразится в приложении и сотрудник свяжется с вами по номеру телефона для подтверждения заявки.

2.4.3 Структурная схема приложения

Структурная схема приложения – это таблица, которая отображает компоненты и их возможности в приложении. Обычно она используется для визуализации архитектуры программного обеспечения и показывает, как различные модули приложения реализованы.

Таблица 3 «Структурная схема формы номер 1»

Форма 1 - Управление записями	Функциональные возможности и назначение
DataGridView1 – Записи на приём	Отображение таблицы из базы данных
TextBox1 – ID записи	Отображение соответствующей ячейки в базе данных
TextBox2 – Дата внесения записи	Отображение соответствующей ячейки в базе данных
TextBox3 – Дата и время записи на приём	Отображение соответствующей ячейки в базе данных
TextBox4 – ФИО	Отображение соответствующей ячейки в базе данных
TextBox5 – Телефон	Отображение соответствующей ячейки в базе данных
TextBox6 – Направление обращения	Отображение соответствующей ячейки в базе данных
TextBox7 – Примечания	Отображение соответствующей ячейки в базе данных
Button1 – Удалить	Удаление выбранной строки в элементе DataGridView
Button2 – Сохранить	Сохраняет действия связанные с DataGridView1
Button3 – Добавить	Переход на форму №2
Button4 – О приложении	Переход на форму №3
Button5 – Закрывать	Выключает приложение
DateTimePicker1 – Календарь	Отображает текущую дату
Label1 – Время	Отображает текущее время
ToolStripMenuItemButton1 – Файл	Отображает кнопку ToolStripMenuItemButton2
ToolStripMenuItemButton2 – Экспорт в Excel	Экспортирует данные DataGridView1 в XLSX файл
ToolStripMenuItemButton3 – Поиск	Отображает toolStripTextBox1
ToolStripTextBox1 – Строка поиска	При написании символов ищет совпадения в реальном времени в колонке FIO элемента DataGridView1

Таблица 4 «Структурная схема формы номер 2»

Форма 2 - Добавление записи	Функциональные возможности и назначение
TextBox2 – Дата внесения записи	Ввод данных
TextBox4 – ФИО	Ввод данных

TextBox5 – Телефон	Ввод данных
TextBox6 – Направление обращения	Ввод данных
TextBox7 – Примечания	Ввод данных
Button2 – Назад	Переход на форму №1
Button3 – Добавить	Добавляет новую запись (строку) в DataGridView1
TextBox4 – ФИО	Ввод данных

Таблица 5 «Структурная схема формы номер 3»

Форма 3 - О приложении	Функциональные возможности и назначение
Button2 – Назад	Переход на форму №1

2.5. Описание структуры входной и выходной информации

Входную информацию пользователь вносит информацию в таблицы, которые находятся в базе данных, а также вводя запросы в базе данных с помощью запросов системы управления базами данных PostgreSQL.

Выходной информацией будут таблицы, заполненные ранее и результаты выполнения запросов.

Входной информацией в разрабатываемом программном средстве будет служить следующая информация:

1. Идентификатор записи;
2. Дата записи;
3. Назначенное время на приём;
4. ФИО гражданина;
5. Мобильный номер телефона гражданина;
6. Цель визита и номер кабинета, который планирует посетить гражданин;
7. Примечание (при наличии).

Таблица 6 «Описание входной и выходной информации в табличной форме»

№ п/п	Входная и выходная информация	Описание
-------	-------------------------------	----------

1	Идентификатор записи	Номер записи в базе данных (заполняется автоматически)
2	Дата записи	Дата внесения записи в базу данных
3	Назначенное время на приём	Назначенная дата и время, когда гражданин должен явиться на приём
4	ФИО гражданина	Фамилия, Имя и Отчество гражданина, которому нужна социальная помощь
5	Мобильный номер телефона гражданина	Мобильный номер телефона гражданина, который записывался на приём в ГКУ “КЦСЗН”
6	Цель визита и номер кабинета, который планирует посетить гражданин	Описание цели визита ГКУ “КЦСЗН” и номер кабинета, в который планирует обратиться гражданин
7	Примечание (при наличии)	Дополнительное описание

2.6. Отладка и тестирование приложения

Тестирование разрабатываемого программного продукта необходимо для подтверждения работоспособности и соответствия выполненных работ в соответствии с техническим заданием прописанным заказчиком или разработчиком.

Требованиями к данному программному обеспечению являются следующие пункты:

1. Корректное отображение выходной информации;
2. Работоспособность кнопок: сохранить, изменить, удалить, сортировка, сброс и сохранить;
3. Работоспособность поиска по ФИО;
4. Работоспособность программного продукта в общем;

2.6.1 Отладка и тестирование разработанного приложения и сайта

Управление записями

Файл	Поиск		IDЗаписи	ДатаЗаписи	ДатаИВремяЗаписиНаПриём	ФИО	НомерМобильногоТелефона	НаправлениеОбращения	Примечание
		▶	4	08.07.2022 19:40:00	09.07.2022 20:00:00	Иванов Иван Иванович	+79998876543	Оформление мед. помощи, 1 кабинет	Нет
			2	08.07.2022 1:40:00	09.07.2022 2:00:00	Васильев Василий Васильевич	+79998876522	Соц. помощь, 5 кабинет	Нет
			1	10.08.2022 8:20:02	18.06.2022 10:00:00	Степанов Степан Степанович	+7(991)543-67-89	Социальная помощь 5 кабинет	Нет примечаний
			3	30.01.2020 11:32:00	01.02.2020 11:00:00	Игнатьев Игнат Игнатович	+78882828282	Соц. помощь, 5 кабинет	Нет

ID записи
Дата внесения записи
Дата и время записи на приём
ФИО
Телефон
Направление обращения
Примечания

Добавить
Сохранить
Удалить

14:03:33 18 июня 2023 г.

Закрыть ?

Рисунок 7 – Форма номер 1 «Управление записями»

На форме номер 1 проверке подлежат такие элементы как:

1. Кнопки в количестве 5 штук;
2. Label1 который отображает время;
3. ToolStripTextBox1 который выполняет поиск;
4. ToolStripMenuItemButton2 элемент выполняющий экспорт в excel;
5. Корректность отображаемой информации.

Данные элементы проверяются на работоспособность и выполнение всех поставленных им задач.

Рисунок 8 – Форма номер 2 «Добавление записи»

Элементы, подлежащие проверке на форме номер 2, включают в себя:

1. Две кнопки;
2. Корректность отображаемой информации.

Все эти элементы проверяются на работоспособность и соответствие поставленным задачам.

Рисунок 9 – Форма номер 3 «О приложении»

На третьей форме следует проверить следующие элементы:

1. Кнопку «Назад»;
2. Корректность отображаемой информации.

Эти элементы подлежат проверке на работоспособность и соответствие всем своим поставленным задачам.

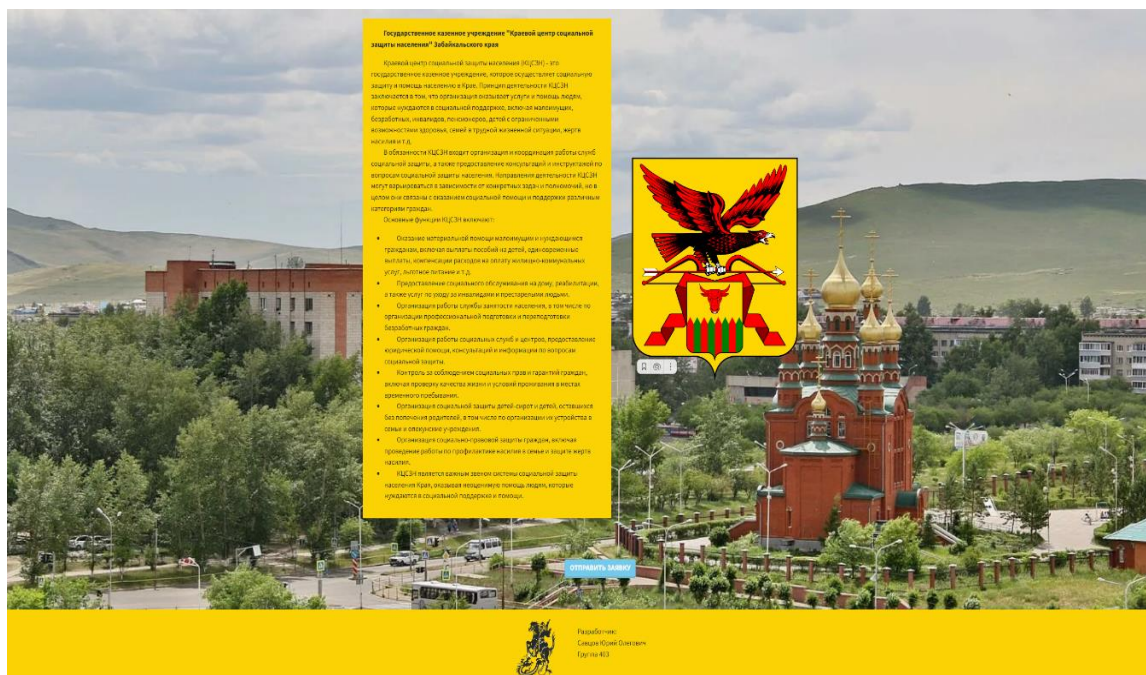
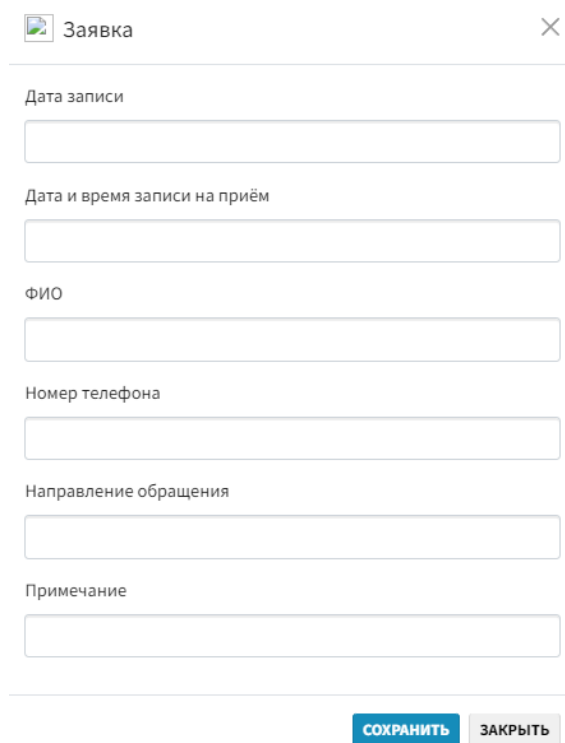


Рисунок 10 – Главная страница сайта

На главной странице сайта проверяется работоспособность таких элементов как:

1. Кнопка «Отправить заявку»;
2. Проверка правильности отображения информации.

Вышеуказанные элементы проходят проверку на работоспособность и соответствие своим поставленным задачам.



Заявка

Дата записи

Дата и время записи на приём

ФИО

Номер телефона

Направление обращения

Примечание

СОХРАНИТЬ ЗАКРЫТЬ

Рисунок 11 – Всплывающее окно на сайте

На всплывающем окне сайта проверяется работоспособность таких элементов как:

1. Две кнопки;
2. Корректность отображаемой информации.

Все эти элементы проверяются на работоспособность и соответствие поставленным задачам.

2.7. Публикация приложения на хостинге

Публикация приложений на хостингах может быть разной в зависимости от того, какой хостинг вы выбрали и какие требования он предъявляет. Однако, в общем случае, процесс публикации приложения на хостинговой платформе можно разделить на следующие шаги:

1. Регистрация на хостинге. Перед тем, как начать публикацию приложения, вам нужно зарегистрироваться на хостинге, который вы выбрали.

Обычно это требует заполнения простой формы с указанием вашего имени, электронной почты, пароля и других данных.

2. Загрузка файлов. После регистрации вам нужно загрузить файлы вашего приложения на хостинг. Это может быть сделано через FTP-клиент или через панель управления хостингом.

3. Настройка хостинга. После загрузки файлов вам может потребоваться настроить некоторые параметры хостинга, такие как настройки базы данных, настройки доступа к файлам и т. д.

4. Тестирование. После того, как вы загрузили файлы и настроили хостинг, можно приступить к тестированию вашего приложения. Убедитесь, что все работает правильно и нет ошибок.

5. Запуск приложения. Если все тесты прошли успешно, то вы можете запустить ваше приложение на хостинге. Обычно для этого нужно настроить параметры запуска и настроить доступ к вашему приложению.

6. Обновление приложения. Если ваше приложение будет обновляться, то вам нужно будет регулярно обновлять его на хостинге. Для этого нужно будет загружать новые файлы и обновлять настройки хостинга.

В целом, процесс публикации приложений на хостинге может быть разным в зависимости от хостинга и требований, которые он предъявляет к приложениям. Но основные шаги остаются одинаковыми - регистрация на хостинге, загрузка файлов, настройка хостинга, тестирование и запуск приложения.

GitHub – это платформа для хранения и совместной разработки программного обеспечения, которая позволяет разработчикам создавать, управлять и делиться своими проектами. Хостинг на GitHub предоставляет множество преимуществ, включая:

1. Бесплатная регистрация и использование: GitHub не взимает плату за

регистрацию или использование хостинга. Это делает его доступным для всех, кто хочет использовать его для своих проектов.

2. Гибкость и масштабируемость: GitHub позволяет настраивать свои проекты в соответствии с вашими потребностями. Вы можете выбирать между различными типами репозиторий, такими как Git, Mercurial и другие, и настраивать их в соответствии с вашим проектом. Кроме того, GitHub предоставляет возможность масштабирования ваших проектов с помощью облачных сервисов.

3. Безопасность: GitHub обеспечивает высокую степень безопасности для ваших проектов. Он использует двухфакторную аутентификацию, которая требует дополнительного подтверждения при входе в систему, а также шифрование данных.

4. Интеграция с другими сервисами: GitHub интегрируется с множеством других сервисов, таких как GitLab, Bitbucket, Trello и др. Это позволяет вам легко обмениваться данными и работать с коллегами в рамках одной платформы.

5. Сообщество: GitHub имеет активное сообщество разработчиков, которые делятся своими знаниями и опытом в области разработки программного обеспечения. Вы можете присоединиться к сообществу и получить поддержку от опытных разработчиков.

В целом, хостинг на GitHub является отличным выбором для разработчиков, которые хотят создать и поддерживать свои проекты. Он предоставляет множество возможностей для совместной работы, обмена данными и поддержки сообщества.

Данная работа хранится на GitHub по ссылке:
<https://github.com/SAS288/Diplom>

2.8. Разработка руководства по использованию приложения

В соответствии с ГОСТ 19.505–79 "Руководство оператора" перед внедрением приложения должно быть написано руководство пользователя.

Руководство по использованию приложения – это документ, который описывает процесс использования приложения и помогает пользователям быстро и эффективно начать работу с ним.

Руководство по использованию приложения должно быть простым и понятным для пользователей, содержать только необходимую информацию и не перегружать ее лишними деталями.

Руководство по использованию приложения состоит из следующих пунктов:

1. Введение;

- 1.1 Краткое описание приложения и его назначение;

- 1.2 Преимущества использования приложения перед традиционным способом записи на прием;

2. Запись на прием;

- 2.1 Пошаговая инструкция по записи на прием к специалисту;

- 2.2 Выбор даты и времени приема;

- 2.3 Подтверждение записи с помощью SMS-кода;

3. Отмена записи на прием;

- 3.1 Возможность отмены записи на прием, если пользователь передумал;

- 3.2 Инструкция для отмены записи;

4. Контакты;

- 4.1 Контактная информация для связи с сотрудниками центра;

- 4.2 Возможность отправки сообщений в службу поддержки;

5. Заключение;

5.1 Подведение итогов;

5.2 Благодарность за использование приложения.

2.8.1 Введение по руководству по использованию приложения

Данное приложение разработано для сотрудников государственного казенного учреждения "Краевой центр социальной защиты населения" с целью повышения эффективности работы и улучшения качества обслуживания граждан.

Создаваемое приложение поможет сократить время ожидания в очереди на прием, что особенно важно для людей с ограниченными возможностями здоровья, пенсионеров и других категорий граждан, которые нуждаются в социальной поддержке.

Приложение будет предоставлять актуальную информацию о графике работы центра, что позволит гражданам спланировать свой визит заранее и избежать необходимости ожидать в очереди.

Приложение ускорит процедуру записи на прием, поскольку граждане смогут выбрать удобное время и дату визита в центр.

2.8.2 Установка и описание основных действий для работы

1. Установка приложения:

Загрузите установочный файл приложения для записи на прием в государственное казенное учреждение "Краевой центр социальной защиты населения";

Запустите файл и следуйте инструкциям на экране, чтобы установить приложение на ваш компьютер.

2. Запись на прием:

Примите звонок гражданина (приложение разрабатывается для

сотрудников);

Если гражданин желает записаться на приём запустите приложение;

Добавление записи происходит при нажатии кнопки “Добавить”;

Запишите данные согласно подписанным полям в приложении;

3. Отмена записи:

Если гражданин не может посетить прием в назначенное время он позвонит сотруднику скажет дату и время, на которое назначался приём, и сотрудник удалит или перенесёт запись в приложении;

Отмена записи происходит при выборе строки в таблице определённого гражданина и нажатии кнопки “Удалить”;

Перенос записи происходит при изменении строки в таблице определённого гражданина и нажатии кнопки “Изменить”.

2.8.3 Заключение по руководству по использованию приложения

Программный комплекс соответствует всем заявленным требованиям и полностью соответствует стандартам разработки и внедрения программного обеспечения.

Благодаря проведенному тестированию и исправлению ошибок удалось достичь высокой производительности и надежности программного обеспечения. Таким образом, созданное программное обеспечение будет полезно для государственного казенного учреждения "Краевой центр социальной защиты населения" и поможет повысить эффективность работы.

Благодарю вас за использование приложения! Я надеюсь, что мое приложение было для вас полезным и помогло решить ваши задачи. Если у вас есть какие-либо отзывы или предложения по улучшению работы приложения, пожалуйста, сообщите мне. Я всегда готов улучшать свои навыки и сделать мое приложение еще лучше!

ЗАКЛЮЧЕНИЕ

Данная дипломная работа по теме «Разработка информационной системы записи на приём в ГКУ КЦСЗН» была выполнена на двух программах для программирования Visual Studio, Visual Studio Code и системе управления базами данных PostgreSQL.

При выполнении данной работы были использованы такие языки программирования как мультипарадигменный язык программирования JavaScript, TypeScript язык, расширяющий возможности JavaScript, язык таблиц стилей CSS, язык разметки HTML, объектно-ориентированный язык программирования C#, декларативный язык программирования SQL.

В ходе выполнения работы были выполнены поставленные задачи:

1. Провести анализ предметной области;
2. Рассмотреть аналоги создаваемой программы и их характеристики;
3. Выбрать средства разработки программного обеспечения;
4. Разработать информационно-логическую модель приложения;
5. Разработать требования к программному средству;
6. Разработать пользовательский интерфейс;
7. Спроектировать и создать базу данных;
8. Протестировать на работу программы;
9. Разработать руководство по использованию программы.

Приложение, согласно техническому заданию, выполняет все поставленные задачи.

В ходе разработки программного продукта полученные знания во время учебных занятий были закреплены на практике.

Для выполнения работы были рассмотрены такие задачи, как изучение аналогов программного обеспечения, теоретических материалов, построение задач и алгоритмы блок-схем, проектирование программного обеспечения и баз данных, написание программного кода, а также тестирование и отладка ПО.

Каждая из этих задач была выполнена полностью.

В процессе написания программного кода были закреплены полученные знания на практике.

ЛИТЕРАТУРА

Законодательные и нормативные акты:

1. ГОСТ Р 7.0.12-2011 Библиографическая запись. Сокращение слов и словосочетаний на русском языке. Общие требования и правила. – М.: Стандартинформ, 2012. – 61 с.
2. ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: Стандартинформ, 2010. – 92 с.
3. ГОСТ 7.32-2017 Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: Стандартинформ, 2017. – 47 с.
4. ГОСТ 7.82-2001 Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления. – М.: ИПК Издательство стандартов, 2001. – 39 с.
5. ГОСТ Р 7.0.100-2018 Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: Стандартинформ, 2018. – 122 с.
6. ГОСТ Р 7.0.5-2008 Библиографическая ссылка. Общие требования и правила составления. – М.: Стандартинформ, 2008. – 32 с.
7. Единая система программной документации. – М.: Стандартинформ, 2005. – 128 с.

Учебная и научная литература:

1. Иванова, Г.С. Технология программирования: учебник для студентов вузов обуч. по напр. «Информатика и вычислительная техника» / Г.С. Иванова. 3-е изд., стер. – Москва: Кнорус, 2018. – 333 с.
2. Павловская, Т.А. С#. Программирование на языке высокого уровня: учебник для студентов вузов. – СПб: Питер, 2020. – 432 с.

3. Перлова, О.Н., Ляпина, О.П., Гусева, А.В. Проектирование и разработка информационных систем: учебник. – 2-е изд, стер. – М.: Издательский центр «Академия», 2018. – 256 с.
4. Федорова, Г.Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности: Учебное пособие / Г.Н. Федорова. – М.: КУРС: ИНФРА-М, 2019. – 336 с.
5. Фуфаев, Д.Э., Фуфаев, Э.В., Разработка и эксплуатация автоматизированных информационных систем: учебное пособие для студентов учреждений среднего профессионального образования по специальности "Информатика и вычислительная техника" / Д.Э. Фуфаев, Э.В. Фуфаев. – 6-е изд., стер. – М.: Издательский центр «Академия», 2018. – 302 с.

Интернет-документы:

1. Введение в язык C# и .NET Framework. – [Электронный ресурс]. – URL: (дата обращения: 25.09.2021).
2. Гагарина, Л. Г. Технология разработки программного обеспечения: учебное пособие / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Сидорова-Виснадул; под ред. Л.Г. Гагариной. – Москва: ФОРУМ: ИНФРА-М, 2020. – 400 с. – (Среднее профессиональное образование). – ISBN 978-5-8199-0812-9. – Текст: электронный. – URL: <https://znanium.com/catalog/product/1067012> (дата обращения: 30.09.2021).
3. Гниденко, И. Г. Технологии и методы программирования: учебное пособие для вузов / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. – Москва: Издательство Юрайт, 2020. – 235 с. – (Высшее образование). – ISBN 978-5-534-02816-4. – Текст : электронный // ЭБС Юрайт [сайт]. – URL: <https://urait.ru/bcode/450999> (дата обращения: 25.09.2021).
4. Гниденко, И. Г. Технология разработки программного обеспечения: учебное пособие для среднего профессионального образования /

И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. – Москва: Издательство Юрайт, 2020. – 235 с. – (Профессиональное образование). – ISBN 978-5-534-05047-9. – Текст: электронный // ЭБС Юрайт [сайт]. – URL: <https://urait.ru/bcode/453640> (дата обращения: 26.09.2021).

5. Гуриков, С. Р. Введение в программирование на языке Visual C#: учебное пособие / С.Р. Гуриков. — МОСКВА: ФОРУМ: ИНФРА-М, 2020. — 447 с. — (Высшее образование: Бакалавриат). — ISBN 978-5-00091-458-8. — Текст: электронный. — URL: <https://znanium.com/catalog/product/1092167> (дата обращения: 29.09.2021).

6. Интернет-сервис для построения схем и диаграмм Draw.io. – [Электронный ресурс]. – URL: <https://www.draw.io/> (дата обращения: 30.09.2021).

7. Интернет-сервис для построения UML-диаграмм. – [Электронный ресурс]. – URL: <https://plantuml.com/> (дата обращения: 30.09.2021).

8. Казанский, А. А. Программирование на Visual C#: учебное пособие для вузов / А. А. Казанский. – 2-е изд., перераб. и доп. – Москва: Издательство Юрайт, 2020. – 192 с. – (Высшее образование). – ISBN 978-5-534-12338-8. – Текст: электронный // ЭБС Юрайт [сайт]. – URL: <https://urait.ru/bcode/451467> (дата обращения: 30.09.2021).

9. Полное руководство по языку программирования C# 6.0 и платформе

10. .NET 4.6. – [Электронный ресурс]. – URL: <http://metanit.com/sharp/tutorial/> (дата обращения: 27.09.2021).

11. Руководство по программированию в Windows Forms. – [Электронный ресурс]. – URL: <http://metanit.com/sharp/windowsforms/> (дата обращения: 27.09.2021).

12. Руководство по программированию в WPF. – [Электронный ресурс].

URL: <https://metanit.com/sharp/wpf/> (дата обращения: 29.09.2021).

Руководство по работе в среде Visual Studio. – [Электронный ресурс].

URL: <https://docs.microsoft.com/ru-ru/visualstudio/> (дата обращения:
25.09.2021).

ПРИЛОЖЕНИЕ №1 «ЛИСТИНГ ПРОГРАММЫ»

Листинг приложения

Форма №1

```
using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using Npgsql;

using System.Data.SqlClient;

using Excel = Microsoft.Office.Interop.Excel;

namespace Project_1
{
    enum RowState
    {
        New,
        Modified,
        ModifiedNew,
    }
}
```

```

        Existed,

        Deleted
    }

public partial class Form1 : Form
{

    Timer timer = new Timer();

    DataBase db = new DataBase();

    int selectedRow;

    public Form1()
    {
        InitializeComponent();
    }

    private void CreateColumns()
    {
        dataGridView1.Columns.Add("id", "ID записи");
        dataGridView1.Columns.Add("dvz", "Дата записи");
        dataGridView1.Columns.Add("divznp", "Дата и время записи на приём");
        dataGridView1.Columns.Add("fio", "ФИО");
        dataGridView1.Columns.Add("tel", "Номер мобильного телефона");
    }

```

```

        dataGridView1.Columns.Add("nap_ob", "Направление обращения");
        dataGridView1.Columns.Add("ptim", "Примечание");
        dataGridView1.Columns.Add("IsNew", String.Empty);
    }

    private void ReadSingRow(DataGridView dgw, IDataRecord record)
    {
        dgw.Rows.Add(record.GetInt32(0), record.GetDateTime(1),
        record.GetDateTime(2), record.GetString(3), record.GetString(4),
        record.GetString(5), record.GetString(6), RowState.ModifiedNew);
    }

    private void RefreshDataGrid(DataGridView dgw)
    {
        dgw.Rows.Clear();

        string queryString = $"select * from zapis_na_priyoms";

        NpgsqlCommand command = new NpgsqlCommand(queryString,
        db.GetConnection());

        db.OpenConnection();

        NpgsqlDataReader reader = command.ExecuteReader();

        while (reader.Read())

```

```

    {
        ReadSingRow(dgw, reader);
    }

    reader.Close();
}

private void deleteRow()
{
    int index = dataGridView1.CurrentCell.RowIndex;

    dataGridView1.Rows[index].Visible = false;

    if (dataGridView1.Rows[index].Cells[0].Value.ToString() == string.Empty)
    {
        dataGridView1.Rows[index].Cells[7].Value = RowState.Deleted;
        return;
    }

    dataGridView1.Rows[index].Cells[7].Value = RowState.Deleted;
}

private void update()
{
    db.OpenConnection();

    for (int index = 0; index < dataGridView1.Rows.Count; index++)

```

```

{
    var rowState = (RowState)dataGridView1.Rows[index].Cells[7].Value;

    if (rowState == RowState.Existed)
    {
        continue;
    }

    if (rowState == RowState.Deleted)
    {
        var id = Convert.ToInt32(dataGridView1.Rows[index].Cells[0].Value);
        var deletedQuery = $" delete from zapis_na_priyoms where id = {id}";

        var command = new NpgsqlCommand(deletedQuery,
db.GetConnection());

        command.ExecuteNonQuery();
    }
}

db.CloseConnection();

RefreshDataGrid(dataGridView1);
}

private void Form1_Load(object sender, EventArgs e)

```

```

{

    timer.Interval = 1000;

    timer.Tick += new EventHandler(timer1_Tick);

    timer.Start();

    CreateColumns();

    RefreshDataGrid(dataGridView1);
}

private void timer1_Tick(object sender, EventArgs e)
{
    int h = DateTime.Now.Hour;

    int m = DateTime.Now.Minute;

    int s = DateTime.Now.Second;

    string time = "";

    if (h < 10)
    {
        time += "0" + h;
    }
    else
    {
        time += h;
    }
}

```

```
}
```

```
time += ":";
```

```
if (m < 10)
```

```
{
```

```
    time += "0" + m;
```

```
}
```

```
else
```

```
{
```

```
    time += m;
```

```
}
```

```
time += ":";
```

```
if (s < 10)
```

```
{
```

```
    time += "0" + s;
```

```
}
```

```
else
```

```
{
```

```
    time += s;
```

```
}
```

```
label1.Text = DateTime.Now.ToLongTimeString();
```

```
}
```

```
private void dataGridView1_CellClick(object sender,  
DataGridViewCellEventArgs e)
```

```
{
```

```
    selectedRow = e.RowIndex;
```

```
    if (e.RowIndex >= 0)
```

```
    {
```

```
        DataGridViewRow row = dataGridView1.Rows[selectedRow];
```

```
        textBox1.Text = row.Cells[0].Value.ToString();
```

```
        textBox2.Text = row.Cells[1].Value.ToString();
```

```
        textBox3.Text = row.Cells[2].Value.ToString();
```

```
        textBox4.Text = row.Cells[3].Value.ToString();
```

```
        textBox5.Text = row.Cells[4].Value.ToString();
```

```
        textBox6.Text = row.Cells[5].Value.ToString();
```

```
        textBox7.Text = row.Cells[6].Value.ToString();
```

```
    }
```

```
}
```

```
private void button1_Click(object sender, EventArgs e)
```

```
{
```

```
    deleteRow();
```

```
}
```



```

private void button2_Click(object sender, EventArgs e)
{
    if (textBox1.Text != "" && textBox2.Text != "" && textBox3.Text != "" &&
        textBox4.Text != "" && textBox5.Text != "" && textBox6.Text != "" &&
        textBox7.Text != "")
    {
        string cmd_text = "update zapis_na_priyoms set dvz = " + textBox2.Text +
            ", " +
            "divznp = " + textBox3.Text + ", " +
            "fio = " + textBox4.Text + ", " +
            "tel = " + textBox5.Text + ", " +
            "nap_ob = " + textBox6.Text + ", " +
            "ptim = " + textBox7.Text + " where id = " + textBox1.Text + """;

        NpgsqlCommand command = new NpgsqlCommand(cmd_text,
            db.GetConnection());

        db.OpenConnection();

        command.ExecuteNonQuery();

        db.CloseConnection();

        RefreshDataGrid(dataGridView1);
    }
    else

```

```

    {
        MessageBox.Show("Ошибка");
    }
}

private void toolStripTextBox1_TextChanged(object sender, EventArgs e)
{
    try
    {
        string select = "Select * from zapis_na_priyoms where fio Like '" +
toolStripTextBox1.Text + "%'";

        db.OpenConnection();

        NpgsqlCommand command = new NpgsqlCommand(select,
db.GetConnection());

        NpgsqlDataAdapter dataAdapter = new NpgsqlDataAdapter(command);
        DataTable dataTable = new DataTable();

        NpgsqlDataReader reader = command.ExecuteReader();

        while (reader.Read())
        {
            dataGridView1.Rows.Clear();

            ReadSingRow(dataGridView1, reader);
        }
    }
}

```

```

    }

    reader.Close();

    if (toolStripTextBox1.Text == "")
    {
        RefreshDataGrid(dataGridView1);
    }
}
catch
{
}
}

private void button3_Click(object sender, EventArgs e)
{
    Form2 form2 = new Form2();
    form2.ShowDialog();
}

private void button4_Click(object sender, EventArgs e)
{
    Form3 form3 = new Form3();
    form3.ShowDialog();
}

```

```

private void ToolStripMenuItemButton2_Click(object sender, EventArgs e)
{
    Excel.Application exApp = new Excel.Application();

    exApp.Workbooks.Add();

    Excel.Worksheet ws = (Excel.Worksheet)exApp.ActiveSheet;

    int i, j;

    for (i = 0; i <= dataGridView1.RowCount - 2; i++)
    {
        for (j = 0; j <= dataGridView1.ColumnCount - 1; j++)
        {
            ws.Cells[i + 1, j + 1] = dataGridView1[j, i].Value.ToString();
        }
    }

    exApp.Visible = true;
}

private void button5_Click(object sender, EventArgs e)
{
    Close();
}
}
}

```

Форма №2

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Npgsql;

namespace Project_1
{
    public partial class Form2 : Form
    {
        DataBase db = new DataBase();
        public Form2()
        {
            InitializeComponent();
        }
    }
}
```

```

private void button2_Click(object sender, EventArgs e)
{
    this.Hide();
}

private void button3_Click(object sender, EventArgs e)
{
    db.OpenConnection();

    var dvz = textBox2.Text;
    var divznp = textBox3.Text;
    var fio = textBox4.Text;
    var tel = textBox5.Text;
    var nap_ob = textBox6.Text;
    var ptim = textBox7.Text;

    var addQuery = $" insert into zapis_na_priyoms (dvz, divznp, fio, tel, nap_ob,
ptim)" +
    $" values ('{dvz}', '{divznp}', '{fio}', '{tel}', '{nap_ob}', '{ptim}')";

    var command = new NpgsqlCommand(addQuery, db.GetConnection());
    command.ExecuteNonQuery();

    MessageBox.Show("Запись сохранена ");
}

```

```

        db.CloseConnection();
    }
}

```

Форма №3

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

```

```

namespace Project_1
{
    public partial class Form3 : Form
    {
        public Form3()
        {
            InitializeComponent();
        }
    }
}

```

```
private void button2_Click(object sender, EventArgs e)
{
    this.Hide();
}
}
```