МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки)

Разработка программного обеспечения
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка программного обеспечения информационной системы поддержки принятия решений оператора сотовой связи

Обучающийся Н.И. Шабалин _____
(Инициалы Фамилия) (личная подпись)

Руководитель канд. техн. наук Н.В. Хрипунов
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

Выпускная квалификационная работа (ВКР) посвящена разработке информационной системы поддержки принятия решений для операторов сотовой связи.

Работа содержит введение, основную часть, представленную четырьмя главами, заключение и список используемых источников.

Во введении представлена актуальность выбранной темы ВКР, определены цели и задачи, а также обозначена практическая ценность разрабатываемой информационной системы.

В первой главе выполнен анализ предметной области, включающий изучение особенностей работы колл-центров операторов сотовой связи и выявление существующих проблем и требований.

Во второй главе проведен выбор средств разработки, включая язык программирования и фреймворк, системы управления базами данных и другие инструменты.

В третьей главе описан процесс разработки информационной системы. Разработана архитектура системы, спроектирована база данных и разработан пользовательский интерфейс. Также описаны особенности обработки данных и логика перехода между страницами.

В четвертой главе проведено тестирование разработанной системы.

В заключении представлены итоги проектирования и реализации информационной системы. Обобщены результаты работы, оценена эффективность достигнутых целей, а также обозначены перспективы развития и дальнейшие исследования в данной области.

Оглавление

Введение

В наше время большинство людей имеют смартфон, который является неотъемлемой частью повседневной жизни большинства людей. В каждом из этих устройств есть SIM-карта, которая принадлежит какому-либо оператору сотовой связи. На фоне насыщенного рынка сотовой связи, привлечение новых клиентов происходит чаще всего за счет переходов от других операторов. Поэтому важно в таких условиях поддерживать лояльность клиентов, так как лояльный клиент не только остаётся верен компании, но и рекомендует её своим друзьям и близким, что влияет на привлечение новых клиентов, что в свою очередь увеличивает прибыль компании. Но, к сожалению, чаще всего лояльность клиента уменьшается из-за того, что сотрудники колл-центров часто совершают ошибки, что негативно сказывается на уровне обслуживания и, как следствие, на лояльности клиентов. Ошибки могут происходить по разным причинам: невнимательность, отсутствие опыта у новых сотрудников или незнание обновленных процедур обслуживания. Опытные сотрудники, полагаясь на память, могут упустить важные детали, что также замедляет процесс решения проблем клиентов. Все эти факторы подчеркивают необходимость разработки системы, которая поможет операторам принимать правильные решения быстро и эффективно.

В данной работе объектом исследования является информационная система поддержки принятия решений для операторов сотовой связи. Предметом исследования является разработка программного обеспечения, способного улучшить качество и скорость обслуживания клиентов в колл-центре. Целью данной выпускной квалификационной работы является разработка программного обеспечения информационной системы поддержки принятия решений для операторов сотовой связи, которая поможет улучшить процесс обслуживания клиентов в колл-центре. Для достижения данной цели необходимо решить следующие задачи:

– проанализировать существующие системы поддержки принятия решений в колл-центрах;
– спроектировать интерфейс пользователя для программы, включающий окна ввода номера телефона клиента, отображения истории обращений и выбора новой проблемы;
– разработать пошаговую инструкцию по решению выбранной проблемы, включающую вопросы для уточнения и предоставление решений на основе введенных данных;
– реализовать систему на языке программирования Java с использованием фреймворка Spring;
– обеспечить запись информации о новых обращениях и решениях в базу данных;
– провести тестирование разработанной системы и оценить её эффективность.

Во введении описывается актуальность задачи, формулируется цель и ставятся задачи, поставленные в работе.

В первой главе исследуется предметная область и происходит выбор инструментов для создания информационной системы.

Во второй главе исследуются и выбираются средства разработки.

В третей главе описывается разработка информационной системы.

В четвертой главе производится тестирование разработанной информационной системы.

Глава 1 Анализ предметной области и постановка задачи на проектирование

1.1 История развития сотовой связи

История развития сотовой связи охватывает несколько десятилетий, начиная с первых экспериментов беспроводной передачи голоса и данных, заканчивая современными технологиями 5G, которые обеспечивают высокоскоростной доступ к интернету.

С развитием технологий появлялись новые поколения сотовой связи, каждое из которых вносило значительные улучшения. Первое поколение (1G) появилось в 1980-х годах и предоставляло аналоговую голосовую связь. В начале 1990-х годов с внедрением цифровых технологий запустилось второе поколение (2G), которое обеспечивало более лучшее качество связи, увеличивало емкость сетей и добавляло услуги, такие как текстовые сообщения (SMS). Третье поколение (3G), запущенное в начале 2000-х годов, обеспечивало более высокие скорости передачи данных и с помощью 3G можно было уже использовать мобильный интернет и видеозвонки [1]. Четвертое поколение (4G) с технологиями LTE, появившимися в конце 2009 года, обеспечило высокоскоростной доступ к интернету, поддерживая такие ресурсоемкие приложения, как потоковое видео и онлайн-игры. Современное пятое поколение (5G), которое запустилось еще в 2019 году, обеспечивает ещё более высокие скорости передачи данных, низкую задержку и возможность подключения множества устройств одновременно.

Развитие инфраструктуры сотовой связи также играло ключевую роль. Сети сотовой связи состоят из сотен тысяч базовых станций, каждая из которых покрывает определенную территорию. Постоянная модернизация инфраструктуры позволяет поддерживать высокое качество связи и расширять зоны покрытия. Параллельно с развитием инфраструктуры происходило совершенствование мобильных устройств [11]. Переход от

аналоговых телефонов к компактным цифровым устройствам, а затем к современным смартфонам с сенсорными экранами и мощными процессорами изменил способ взаимодействия людей с мобильными сетями. Современные смартфоны стали многофункциональными устройствами, объединяющими в себе функции телефонов, компьютеров, камер и многих других гаджетов [20].

Развитие сотовой связи оказало огромное влияние на общество и экономику [9]. Мобильные устройства стали неотъемлемой частью повседневной жизни, изменив наши способы общения, работы, получения информации и развлечений. Сотовая связь сыграла ключевую роль в глобализации, предоставив людям возможность поддерживать связь независимо от их местоположения [6].

1.2 Современные технологии в колл-центрах

Современные технологии играют ключевую роль в повышении эффективности и качества работы колл-центров. Основные технологии, которые используются в колл-центрах, включают автоматизированные системы обработки вызовов, системы управления взаимоотношениями с клиентами (CRM), искусственный интеллект (AI) и машинное обучение (ML), анализ данных и биг-дата, а также технологии обеспечения безопасности данных.

Автоматизированные системы обработки вызовов, такие как интерактивные голосовые ответы (IVR), позволяют автоматизировать рутинные задачи и сокращать время ожидания клиентов на линии. IVR-системы позволяют клиентам самостоятельно получать необходимую информацию, например, о балансе счета или статусе заказа, без участия оператора. Это сокращает нагрузку на сотрудников и позволяет им сосредоточиться на более сложных задачах.

Системы управления взаимоотношениями с клиентами (CRM) обеспечивают централизованное хранение и управление информацией о клиентах [3]. CRM-системы позволяют сотрудникам колл-центра быстро получать доступ к истории взаимодействий с клиентом, что повышает качество обслуживания и ускоряет процесс решения проблем. Современные CRM-системы также интегрируются с другими платформами и инструментами, обеспечивая единое окно для работы с клиентами через различные каналы связи.

Искусственный интеллект (AI) и машинное обучение (ML) становятся все более важными в колл-центрах. AI и ML позволяют автоматизировать анализ данных и предоставление рекомендаций операторам. Например, чат-боты, основанные на AI, могут обрабатывать простые запросы клиентов в режиме реального времени, что снижает нагрузку на операторов. ML-алгоритмы могут анализировать большие объемы данных и выявлять паттерны, которые помогают улучшить процессы обслуживания и предлагать персонализированные решения для клиентов.

Анализ данных и биг-дата играют важную роль в улучшении качества обслуживания клиентов и оптимизации работы колл-центров. Сбор и анализ данных о взаимодействиях с клиентами позволяет выявлять проблемные области и разрабатывать стратегии их улучшения. Биг-дата также используется для прогнозирования потребностей клиентов и адаптации услуг под их предпочтения. Современные аналитические инструменты предоставляют колл-центрам возможность мониторинга ключевых показателей эффективности (KPI) в режиме реального времени, что позволяет оперативно принимать решения и улучшать качество обслуживания.

Технологии обеспечения безопасности данных являются неотъемлемой частью современных колл-центров. С учетом увеличения количества кибер-угроз, обеспечение безопасности данных клиентов становится одной из

главных задач. Современные системы безопасности включают шифрование данных, многофакторную аутентификацию, мониторинг и предотвращение угроз, а также регулярные аудиты безопасности. Эти меры помогают защитить данные клиентов от несанкционированного доступа и утечек, обеспечивая доверие клиентов к компании.

Таким образом, современные технологии играют ключевую роль в трансформации работы колл-центров, повышая их эффективность, качество обслуживания и безопасность данных. Внедрение этих технологий позволяет компаниям адаптироваться к новым условиям рынка и удовлетворять ожидания клиентов.

1.3 Типы проблем клиентов

Клиенты мобильных операторов связи могут сталкиваться с различными проблемами, которые могут существенно влиять на их опыт и удовлетворенность услугами. Эти проблемы можно классифицировать по нескольким категориям: технические проблемы, проблемы с устройствами, вопросы тарифов и биллинга, проблемы с обслуживанием клиентов, а также проблемы, связанные с мошенничеством и безопасностью.

Технические проблемы являются самой распространенной категорией. Они могут включать проблемы с сетью, такие как сбои в работе сети или плохое качество сигнала, которое может быть вызвано различными факторами, включая географические особенности местности, перегрузка сети или авария на базовой станции. Также сюда относятся проблемы с SIM-картами, такие как поврежденные или неправильно установленные SIM-карты. Еще одна распространенная техническая проблема, это сбои в работе интернет-сервиса, которые могут быть вызваны проблемами с настройками устройства или с самим интернет-подключением.

Проблемы с устройствами могут быть как аппаратные, так и программные. Аппаратные проблемы, это повреждения экрана, проблемы с

батареей или неисправности кнопок. Программные проблемы могут быть связаны с ошибками в операционной системе, несовместимостью приложений или вирусами.

Вопросы тарифов и биллинга часто являются источником недовольства клиентов. Клиенты могут столкнуться с неожиданными списаниями, неправильными начислениями или непониманием условий тарифного плана или дополнительной услуги. Проблемы могут возникать из-за недостающей или неверной информации о тарифах, ошибок в биллинговой системе или недоразумений по поводу использования услуг. Важно, чтобы операторы колл-центра могли ясно и четко объяснить клиентам структуру тарифов и помочь решить проблемы с биллингом.

Проблемы с обслуживанием клиентов часто связаны с плохим опытом взаимодействия с сотрудниками колл-центра. Это может быть долгое время ожидания на линии или ответа оператора, некомпетентность операторов, невежливость или отсутствие понимания проблемы клиента. Плохое обслуживание клиентов может значительно снизить их лояльность и привести к потере клиентов. Поэтому важно, чтобы операторы были хорошо обучены и имели доступ к необходимым ресурсам для эффективного решения проблем клиентов.

Мошенничество и безопасность также являются важными аспектами, с которыми сталкиваются клиенты. Это могут быть случаи фишинга, когда мошенники пытаются выманить личную информацию клиента, или проблемы с защитой данных, когда конфиденциальная информация клиента может быть скомпрометирована. Клиенты могут также сталкиваться с несанкционированным использованием их аккаунтов или несанкционированными транзакциями. Операторы должны быть готовы предоставить клиентам информацию о том, как защитить свои данные и что делать в случае подозрения на мошенничество.

Таким образом, список проблем с которыми сталкиваются клиенты сотовых операторов, весьма широк. Эффективное решение этих проблем требует хорошо обученного персонала, оснащенного современными технологиями и инструментами для диагностики и устранения неполадок. Разработанная информационная система для поддержки принятия решений, играет ключевую роль в обеспечении высокого уровня обслуживания клиентов, что в свою очередь, способствует повышению их удовлетворенности и лояльности.

1.4 Роль информационных систем в решении проблем клиентов

Информационные системы имеют важную роль в решении проблем клиентов. Они обеспечивают сотрудникам доступ к необходимой информации, автоматизируя рутинные процессы и повышая общую эффективность обслуживания. Основные аспекты, в которых информационные системы способствуют решению проблем клиентов, включают централизованное управление информацией, автоматизацию процессов, анализ данных, персонализацию обслуживания и интеграцию с различными каналами связи.

Централизованное управление информацией является одним из важнейших аспектов информационных систем. Современные CRM-системы позволяют операторам получать доступ к полной истории взаимодействий с клиентом, включая предыдущие обращения и предоставленные решения. Это позволяет операторам быстрее и точнее диагностировать проблему, так как они имеют полное представление о контексте ситуации. Клиент не должен повторно объяснять свою проблему, что значительно сокращает время обслуживания и улучшает клиентский опыт.

Персонализация обслуживания является еще одной ключевой функцией информационных систем. Используя данные о предыдущих взаимодействиях и предпочтениях клиента, системы могут предоставлять

операторам персональные рекомендации и скрипты, которые помогают лучше удовлетворить потребности клиента. Например, если система знает, что клиент предпочитает общение через чат, а не звонок по телефону, то оператор может сразу предложить этот канал для дальнейшего общения. Персональный подход не только улучшает клиентский опыт, но и повышает вероятность успешного разрешения проблемы с первого контакта.

Кроме того, информационные системы играют важную роль в обучении и поддержке операторов. Современные системы предоставляют операторам доступ к базе знаний, где собраны инструкции, решения типовых проблем и ответы на часто задаваемые вопросы. Это помогает новым сотрудникам быстрее адаптироваться и снижает вероятность ошибок.

Таким образом, информационные системы играют незаменимую роль в решении проблем клиентов в колл-центрах. Они обеспечивают централизованное управление информацией, автоматизацию процессов, анализ данных, персонализацию обслуживания и интеграцию с различными каналами связи. Все эти функции способствуют повышению эффективности работы операторов, сокращению времени обслуживания и улучшению общего опыта клиентов, что в свою очередь увеличивает лояльность и удовлетворенность клиентов.

1.5 Преимущества и недостатки существующих решений

Существующие решения, включая CRM-системы и специализированные системы поддержки принятия решений, имеют свои сильные и слабые стороны, которые нужно учитывать при разработке информационной системы поддержки принятия решений для операторов сотовой связи.

Преимущества этих решений заключаются в том, что они обеспечивают структурированное хранение информации о клиентах, их обращениях и истории взаимодействий с компанией. CRM-системы,

например, позволяют сотрудникам легко получать доступ к данным клиента, анализировать их и предоставлять более персонализированный и качественный сервис. Специализированные системы поддержки принятия решений помогают сотрудникам принимать более информированные решения на основе анализа данных и предоставляемых рекомендаций.

Но также у них есть недостатки. Некоторые из решений могут быть сложными в освоении, особенно для новых сотрудников. Это может потребовать дополнительного времени и ресурсов на обучение персонала, что может сказаться на производительности и качестве обслуживания в начальный период работы с системой.

Таким образом, при разработке новой информационной системы для операторов сотовой связи важно учитывать как преимущества, так и недостатки существующих решений, чтобы создать эффективное и гибкое решение, которое соответствует потребностям компании и помогает повысить качество обслуживания клиентов.

1.6 Требования к разрабатываемой системе

Для разработки эффективной информационной системы поддержки принятия решений для операторов сотовой связи необходимо учесть следующие требования:

- удобный и интуитивно понятный интерфейс для операторов колл-центра;
- возможность ввода номера телефона клиента;
- отображение истории обращений клиента с указанием даты, времени, проблемы и предложенного решения;
- пошаговые инструкции по решению проблем с возможностью записи нового обращения в базу данных;

– гибкость системы для адаптации под новые виды проблем и обновления процедур обслуживания;
– надежная и быстрая работа с базой данных для хранения и анализа информации о клиентах и их обращениях.

1.7 Постановка задачи на проектирование

Целью данного проекта является разработка программного обеспечения информационной системы поддержки принятия решений для операторов сотовой связи, которая поможет улучшить процесс обслуживания клиентов в колл-центре. Основные задачи, которые необходимо решить в рамках проекта, включают:

– анализ существующих решений и инструментов для поддержки принятия решений в колл-центрах;
– проектирование интерфейса пользователя, включающего окна для ввода номера телефона, отображения истории обращений и выбора новой проблемы;
– разработка пошаговой инструкции по решению проблем клиентов, включающей вопросы для уточнения и предоставление решений на основе введенных данных;
– обеспечение записи информации о новых обращениях и решениях в базу данных;
– проведение тестирования разработанной системы и оценка её эффективности.

Выводы по главе.

В результате проведенного анализа предметной области выявлена необходимость в разработке информационной системы поддержки принятия решений для операторов сотовой связи. Основные задачи и требования к системе определены, что позволяет приступить к проектированию и

разработке программного обеспечения, которое поможет улучшить качество обслуживания клиентов и повысить их лояльность.

Глава 2 Выбор средств разработки

2.1 Критерии выбора языка программирования высокого уровня

При выборе языка программирования, удовлетворяющего ранее составленным требованиям, необходимо определить критерии, которым он должен соответствовать:

- производительность: учитывая потенциально большой объем данных и высокие требования к отказоустойчивости системы, выбранный язык должен обеспечивать высокую производительность обработки данных;
- расширяемость: главным критерием является возможность обновления функционала в будущем. Язык программирования должен обеспечивать модульность и возможность создания расширяемых компонентов, для обеспечения гибкости при добавлении новых функций;
- безопасность: так как информационная система будет обрабатывать и хранить пользовательские данные, выбранный язык программирования должен поддерживать высокий уровень защиты данных от различного вида атак, таких как SQL-инъекции, CSRF-инъекции.
- поддержка и сообщество: еще одним достаточно важным критерием является наличие активного сообщества у языка программирования, поскольку возможность найти нужную документацию, советы и рекомендации по использованию языка программирования значительно упростит процесс разработки;
- кроссплатформенность: так как существует большое количество операционных систем и браузеров, выбранный язык программирования должен обладать возможностью запуска на различных платформах;

– наличие библиотек и фреймворков: возможность использования в выбранном языке программирования библиотек и фреймворков является важным критерием, так как значительно увеличивает скорость разработки.

2.2 Критерии выбора системы управления базами данных

Необходимо определить критерии для выбора системы управления базами данных:

– производительность: система управления базами данных должна обеспечить высокую производительность и эффективность обработки запросов, чтобы позволить обеспечить быстрый доступ к информации и обработку большого объема данных;
– масштабируемость: система должна иметь возможность горизонтального и вертикального масштабирования для обеспечения корректной работы в условиях увеличения обрабатываемых данных;
– надежность: поскольку информационная система обрабатывает данные пользователей, немаловажным фактором является надежность выбранной системы. Она должна обеспечивать целостность и резервное копирование для предотвращения потери данных;
– безопасность: как и при выборе языка программирования, одним из важных факторов является безопасность выбранной системы для надежного хранения пользовательских данных;
– поддержка и сообщество: немаловажным критерием является наличие активного сообщества у системы управления базами данных, поскольку возможность найти нужную документацию, советы и рекомендации по использованию инструмента значительно

упростит процесс эксплуатации СУБД;

– совместимость: учитывая разнообразие технологий и платформ, выбранная система управления базами данных должна быть совместимой и поддерживать работу с различными операционными системами и языками программирования.

2.3 Рассмотрение языков программирования высокого уровня

Перед тем как сделать окончательный выбор в пользу конкретного языка программирования высокого уровня, рассмотрим тройку самых популярных языков, применимых в разработке: Java, C# и PHP.

В первую очередь рассмотрим язык программирования Java. Будучи одним из самых популярных языков программирования, он предлагает крайне широкий спектр инструментов и библиотек для разработки веб-приложений. [14] Из коробки доступны средства для обеспечения надежности и безопасности, а также масштабирования разрабатываемых систем, что делает Java хорошим выбором для разработки высоконагруженных проектов [2]. В связи с высокой популярностью, язык имеет обширное сообщество разработчиков и отличную техническую поддержку [16].

Теперь рассмотрим язык программирования C#. Обладая мощной средой разработки и широкими возможностями интеграции с такими же продуктами компании, данный язык предоставляет инструменты для разработки как надёжных веб-приложений, так и удобных пользовательских интерфейсов [12].

Последний язык программирования, который мы рассмотрим, это PHP. Он обладает простым синтаксисом, обеспечивает низкий порог вхождения, являясь отличным выбором для начинающих разработчиков, а также, имеет

большое профессиональное сообщество и поддерживает интеграцию со множеством баз данных.

Сравнительный анализ приведённых выше языков программирования представлен в таблице 1.

Таблица 1 – Сравнительный анализ языков программирования

Критерий	Наименование языка программирования		
	Java	C#	PHP
Популярность	высокая	высокая	высокая
Синтаксис	строгая типизация, объектно-ориентированный	строгая типизация, объектно-ориентированный	слабая типизация, скриптовый
Производительность	высокая	высокая	средняя
Поддержка	обширное сообщество, большое количество библиотек	хорошая поддержка от Microsoft, большое количество библиотек	обширное сообщество, множество расширений
Масштабируемость	высокая	высокая	средняя
Безопасность	высокая	высокая	средняя
Интеграция	хорошая поддержка интеграции с различными технологиями	хорошая интеграция с технологиями Microsoft	хорошая интеграция с различными базами данных
Документация	обширная	обширная	обширная

Глубоко проанализировав плюсы и минусы каждого из вариантов, было выбрано решение выбрать язык Java для разработки информационной системы. Его широкие возможности, надежность и масштабируемость подтверждают звание наиболее подходящего языка для реализации поставленных требований [19].

2.4 Рассмотрение систем управления базами данных

Решающую роль в обеспечении эффективного хранения данных играет выбор наиболее подходящей требованиям проекта базы СУБД. Проведем анализ трех одних из самых популярных систем управления базами данных.

Первая СУБД на рассмотрении это PostgreSQL. Будучи мощным open-source решением, широко применяется в индустрии разработки программ. Из отличительных особенностей можно выделить: полная поддержка транзакций и широкий набор функций и типов данных, включая географические данные и временные типы [15].

Теперь рассмотрим MySQL. Это одно из самых популярных решений. Данная СУБД отличается высокой производительностью и простотой использования, при этом предоставляя возможности для масштабирования и репликации данных [8].

Последняя рассматриваемая СУБД, это MongoDB, которая относится к семейству NoSQL, и является не реляционной, как предыдущие две, а документ-ориентированной. Как следствие, данная СУБД поддерживает гибкую модель схемы данных, что позволяет быстро изменять структуру хранимых данных.

Сравнительный анализ приведённых выше систем управления базами данных представлен в таблице 2.

Таблица 2 – Сравнительный анализ систем управления базами данных

Критерий	СУБД		
	PostgreSQL	MySQL	MongoDB
Тип СУБД	реляционная	реляционная	документ-ориентированная
Язык запросов	SQL	SQL	MongoDB Query Language (MQL)
Транзакции	полная поддержка	частичная поддержка	нет
Масштабируемость	отличная	хорошая	отличная
Гибкость схемы данных	структурированная	структурированная	гибкая
Поддержка геоданных	да	ограниченная	да
Средства администрирования	обширные	удобные	удобные
Производительность	высокая	высокая	высокая
Сообщество разработчиков	большое	большое	большое

После рассмотрения самых популярных систем управления базами данных, можно остановиться на PostgreSQL, так как она имеет большое и активное сообщество разработчиков, поддерживает реляционную модель данных, и является полностью бесплатной. PostgreSQL также обладает высокой производительностью и надежностью, что делает ее идеальным выбором. Кроме того, PostgreSQL поддерживает множество расширений, которые могут значительно расширить функциональность базы данных, позволяя разработчикам адаптировать её под конкретные нужды проекта, а также PostgreSQL предлагает широкие возможности для масштабирования и оптимизации производительности.

Выводы по главе.

Подводя итоги, была проведена работа по анализу сильных и слабых сторон важных средств разработки, таких как язык высокого уровня и система управления базами данных. В результате тщательного анализа было принято решение остановиться на связке Java и PostgreSQL, данное сочетание инструментов является надежной и эффективной платформой для разработки и поддержки веб-приложения.

Глава 3 Разработка информационной системы

3.1 Архитектура приложения

Выбор архитектуры при разработке информационной системы поддержки принятия решений для операторов сотовой связи в обеспечении её эффективности, надежности и масштабируемости играет ключевую роль. С использованием фреймворка Spring Boot и шаблонизатора Thymeleaf рассмотрим основные аспекты и преимущества выбранной клиент–серверной архитектуры [18].

3.1.1 Клиент–серверная архитектура

При большом количестве распределенных клиентов отлично помогает контролировать доступ или качество обслуживания шаблон клиент–серверный показанный на рисунке 1.

Рисунок 1 – Схема шаблона клиент–сервер

В подходе «клиент — сервер» соединительные элементы и компоненты обладают характерным поведением:

- компоненты, которые называются «клиентами», отправляют запросы компоненту, который называется «сервер», после чего ждут ответа;
- компонент «сервер» получает запрос от компонента «клиент» после чего отправляет ему ответ.

Клиент–серверный шаблон хорошо использовать при моделировании части системы, имеющих множество компонентов, отправляющих запросы другому компоненту (клиент–сервер), который обеспечивает работу сервисов, — например, онлайн–приложения (электронная почта или обмен документами).

3.1.2 Обоснование выбора архитектуры

Для выбора клиент-серверной архитектуры опираются на следующие факторы:

- гибкость и модульность: с помощью данной архитектуры в клиентской и серверной части можно легко изменять и добавлять новые функциональные возможности;
- масштабируемость и производительность: систему легко масштабировать для обработки большого числа пользователей и запросов от них, а также без проблем справится с обеспечением высокой производительности;
- простота разработки и поддержки: с помощью Spring Boot и Thymeleaf упрощается процесс разработки, тестирования и поддержки системы [7]. Они предоставляют мощные инструменты и интеграции [10];
- безопасность и надежность: Spring Security имеет мощные возможности, которые обеспечивают высокий уровень безопасности данных и доступа [4].

В итоге данная архитектура является надежной, гибкой и масштабируемой основой для разработки информационной системы поддержки принятия решений для операторов сотовой связи.

3.2 Создание концептуальной карты

3.2.1 Сущности

Выделим основные сущности предметной области, они представлены в таблице 3.

Таблица 3 – Список сущностей

Наименование сущности	Краткое описание
client	сущность, представляющая клиента сотовой связи. содержит данные о клиенте, такие как имя, фамилия, номер телефона и уникальный идентификатор.
problem	сущность, представляющая проблему клиента. содержит информацию о проблеме, такую как дата создания проблемы, предложенное решение, заголовок проблемы и уникальный идентификатор.

3.2.2 Концептуальная карта

Концептуальная карта на рисунке 2 показывает графическое представление сущностей информационной системы и их взаимосвязей. В нашем случае сущностями являются клиенты (client) и проблемы (problem), а также есть служебная таблица client_problems, которая устанавливает связь между клиентами и их проблемами.

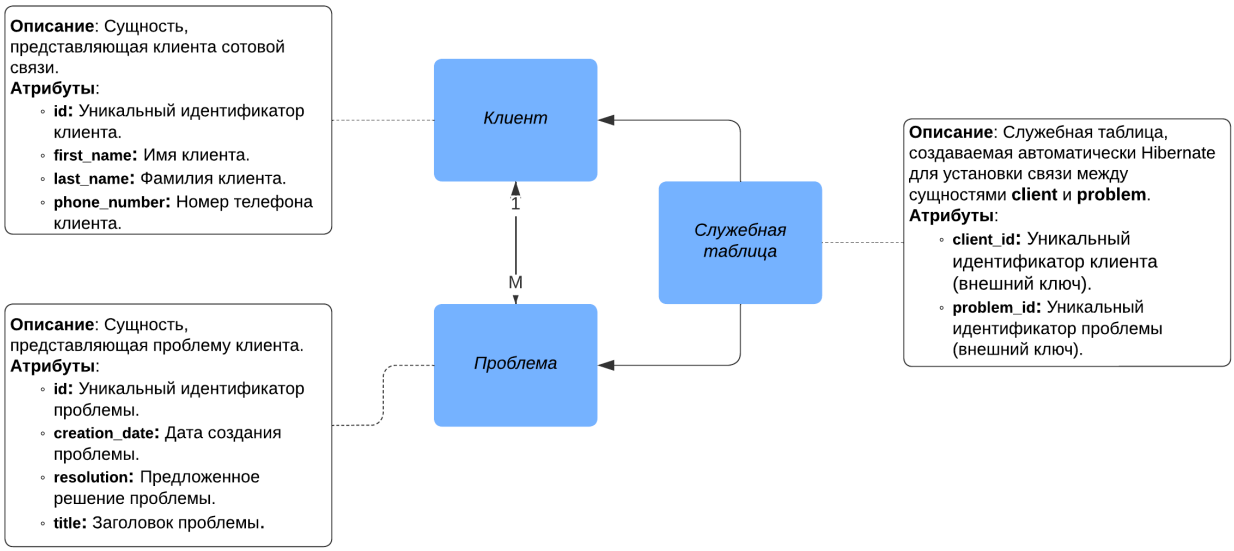


Рисунок 2 – Концептуальная карта

Сущность Client связана с сущностью Problem отношением «один ко многим», что означает, что один клиент может иметь много проблем, но каждая проблема принадлежит только одному клиенту. Эта связь реализована с помощью служебной таблицы Client_Problems.

3.3 Разработка базы данных

Разработка базы данных является ключевым этапом в разработке информационной системы поддержки принятия решений для операторов сотовой связи. Прототип БД помогает понять структуру данных, проверить основные функциональные возможности и убедиться в правильности проектирования перед развертыванием на производственной среде [13].

В качестве системы управления базами данных выбрана PostgreSQL. Этот выбор обусловлен ее надежностью, мощными возможностями работы с данными и поддержкой расширенного SQL.

3.3.1 Сущности и атрибуты

В системе выделены две основные сущности: client и problem. Для связи этих сущностей используется промежуточная таблица client_problems.

а) сущность client:

1) id, уникальный идентификатор клиента (целое число, первичный ключ);
2) first_name, имя клиента (строка);
3) last_name, фамилия клиента (строка);
4) phone_number, номер телефона клиента (строка, уникальный).

б) сущность problem:

1) id, уникальный идентификатор проблемы (целое число, первичный ключ);
2) creation_date, дата и время создания проблемы (timestamp);
3) resolution, предложенное решение проблемы (строка);
4) title, заголовок проблемы (строка).

в) промежуточная таблица client_problems:

1) client_id, идентификатор клиента (целое число, внешний ключ, ссылается на client(id));
2) problem_id, идентификатор проблемы (целое число, внешний ключ, ссылается на problem(id)).

3.3.2 Связи между сущностями

Между сущностями client и problem связь один ко многим через таблицу client_problems. Один клиент может иметь множество проблем. Связь многие ко многим реализована через промежуточную таблицу client_problems, которая позволяет привязывать множество проблем к одному клиенту и наоборот.

3.3.3 Физическая модель

Физическая модель базы данных показана на рисунке 3.

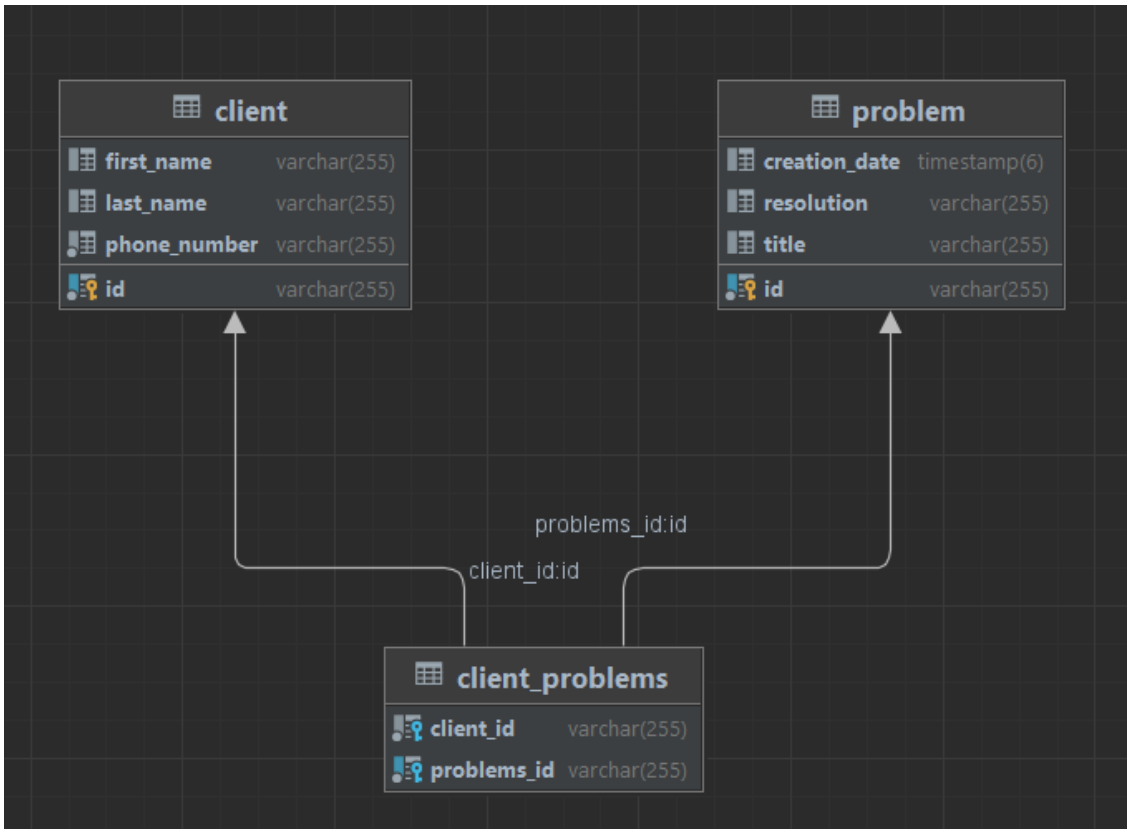


Рисунок 3 – Физическая модель базы данных

3.4 Разработка контроллера

В процессе создания информационной системы поддержки принятия решений оператора сотовой связи был разработан контроллер на языке Java с использованием Spring Framework, код которого показан на рисунке 4. Этот компонент отвечает за обработку запросов, связанных с проверкой номера телефона клиента, и взаимодействие с сервисным слоем для получения информации о клиенте и его проблемах.

Задачи данного компонента включают в себя:

- отображение HTML–страницы для ввода номера телефона клиента;
- обработка запросов на проверку номера телефона;
- получение данных о клиенте из базы данных и передача их на фронтенд для отображения информации о проблемах клиента.

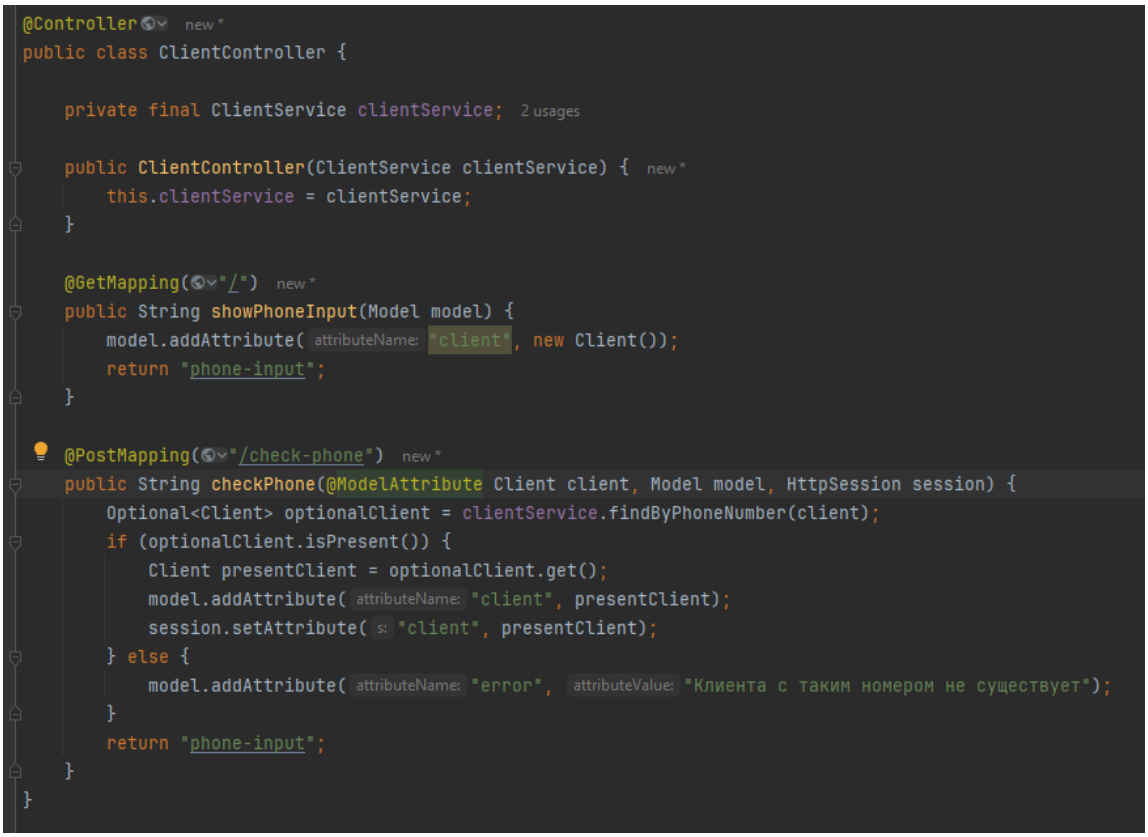


Рисунок 4 – Код контроллера

Описание реализации:

Аннотация @Controller: класс ClientController аннотирован как контроллер Spring MVC, что позволяет ему обрабатывать HTTP–запросы и возвращать HTML–страницы.

Поле clientService: в классе объявлено поле clientService, которое инициализируется через конструктор. ClientService используется для взаимодействия с базой данных и получения информации о клиентах;

Конструктор ClientController: конструктор ClientController (ClientService clientService) позволяет внедрить зависимость ClientService в контроллер.

Метод showPhoneInput:

– аннотирован @GetMapping(«/») для обработки GET–запросов по корневому URL;

– метод добавляет в модель новый объект Client и возвращает название HTML–страницы «phone–input», которая отображается пользователю для ввода номера телефона.

Метод checkPhone:

– аннотирован @PostMapping(«/check–phone») для обработки POST–запросов по URL «/check–phone»;
– метод принимает объект Client из модели с помощью аннотации @ModelAttribute;
– использует clientService для поиска клиента по номеру телефона;
– если клиент найден, данные клиента добавляются в модель и сохраняются в сессии;
– если клиент не найден, в модель добавляется сообщение об ошибке;
– возвращается название HTML–страницы «phone–input» для отображения результата пользователю.

Таким образом, разработанный контроллер является важным компонентом системы поддержки принятия решений оператора сотовой связи. Он обеспечивает функциональность для проверки номера телефона клиента и получения информации о его проблемах, а также помогает сотрудникам колл–центра быстро и эффективно обрабатывать обращения клиентов.

3.4.1 Разработка сущностей

В процессе разработки системы поддержки принятия решений оператора сотовой связи создана функциональность для отображения проблем клиента в пользовательском интерфейсе. Для этого были разработаны сущности Problem и Client, код которых изображен на рисунке 5 и 6 соответственно, а также настроены связи между ними. Эти данные затем отображаются на HTML–странице.

Для реализации данной функциональности были созданы следующие сущности:

Сущность Problem:

- аннотация @Entity указывает, что класс является сущностью JPA и будет отображаться в таблице базы данных;
- таблица называется problem (аннотация @Table(name = «problem»));
- поля включают id (уникальный идентификатор, генерируемый стратегией UUID), title (заголовок проблемы) и resolution (решение проблемы).

```
@Data  4 usages  new *
@Entity
@Table(name = "problem")
@RequiredArgsConstructor
@AllArgsConstructor
public class Problem {

    @Id
    @GeneratedValue(strategy = GenerationType.UUID)
    private String id;

    private String title;

    private String resolution;

    private LocalDateTime creationDate;
}
```

Рисунок 5 – Код сущности Problem

Сущность Client:

- аннотация @Entity указывает, что класс является сущностью JPA и будет отображаться в таблице базы данных;
- таблица называется client (аннотация @Table(name = «client»));
- поля включают id (уникальный идентификатор, генерируемый стратегией UUID), phoneNumber (номер телефона клиента), firstName (имя клиента), lastName (фамилия клиента);

– связь OneToMany указывает, что один клиент может иметь множество проблем. Список проблем хранится в поле problems.

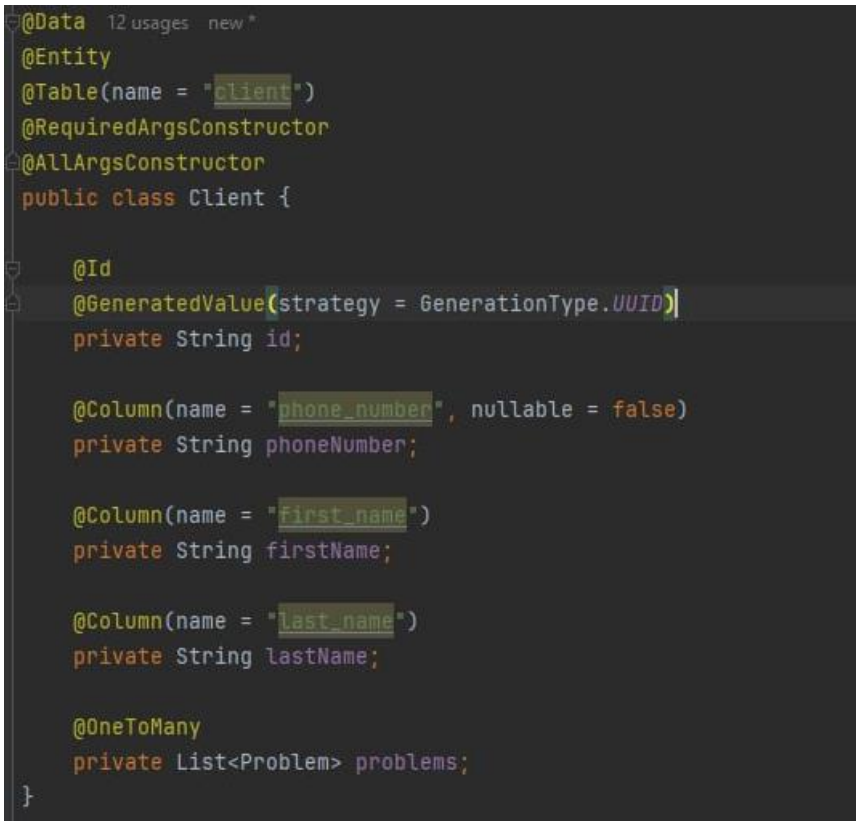


Рисунок 6 – Код сущности Client

Создание и настройка сущностей Problem и Client, а также связи между ними обеспечивает отображение данных на HTML-странице, способствуя эффективному управлению проблемами клиентов.

3.5 Разработка интерфейса конечного пользователя

Для обеспечения удобства работы оператора колл–центра, интерфейс должен быть простым, но функциональным. Именно поэтому разработан интуитивно понятный и удобный интерфейс, чтобы минимизировать количество ошибок при вводе данных и облегчить процесс принятия решений [17].

На начальная странице приложения, изображенной на рисунке 7, можно ввести номер телефона клиента для отображения списка его проблем и добавление новых проблем. Приложение включает следующие основные элементы:

- заголовок страницы: отображает название приложения или основное действие, которое нужно выполнить;
- форма ввода номера телефона: содержит поле для ввода номера телефона клиента и кнопки для отправки формы;
- сообщения об ошибках: отображаются в случае неверного ввода номера телефона;
- список проблем клиента: отображает предыдущие обращения клиента, если таковые имеются.

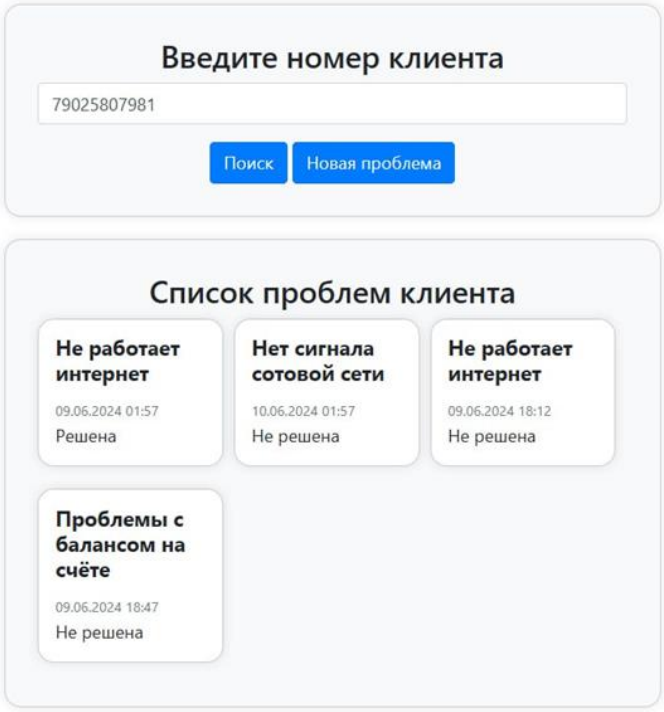
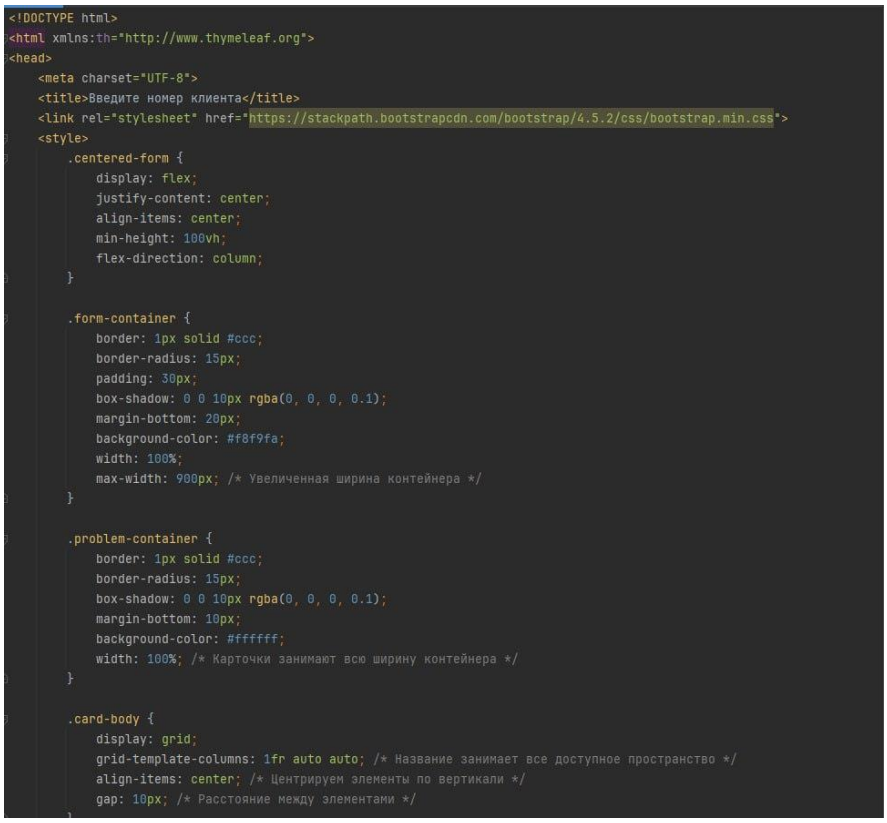


Рисунок 7 – Начальная страница приложения

На рисунках 8 – 11 показан HTML код начальной страницы информационной системы.



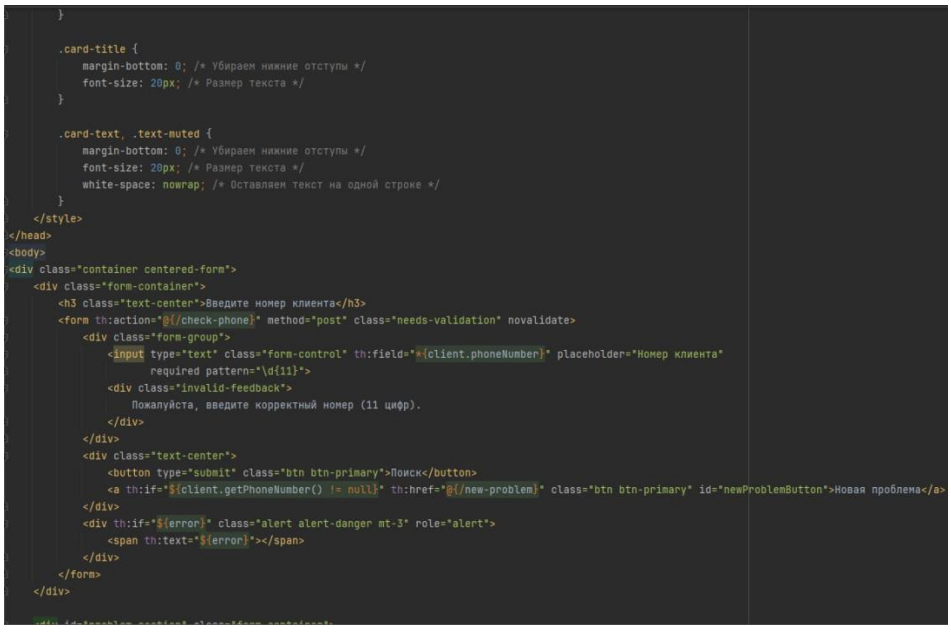
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Введите номер клиента</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <style>
        .centered-form {
            display: flex;
            justify-content: center;
            align-items: center;
            min-height: 100vh;
            flex-direction: column;
        }

        .form-container {
            border: 1px solid #ccc;
            border-radius: 15px;
            padding: 30px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            margin-bottom: 20px;
            background-color: #f8f9fa;
            width: 100%;
            max-width: 900px; /* Увеличенная ширина контейнера */
        }

        .problem-container {
            border: 1px solid #ccc;
            border-radius: 15px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            margin-bottom: 10px;
            background-color: #ffffff;
            width: 100%; /* Карточки занимают всю ширину контейнера */
        }

        .card-body {
            display: grid;
            grid-template-columns: 1fr auto auto; /* Название занимает все доступное пространство */
            align-items: center; /* Центрируем элементы по вертикали */
            gap: 10px; /* Расстояние между элементами */
        }
```

Рисунок 8 – Код начальной страницы, часть 1



```
        }

        .card-title {
            margin-bottom: 0; /* Убираем нижние отступы */
            font-size: 20px; /* Размер текста */
        }

        .card-text, .text-muted {
            margin-bottom: 0; /* Убираем нижние отступы */
            font-size: 20px; /* Размер текста */
            white-space: nowrap; /* Оставляем текст на одной строке */
        }
    </style>
</head>
<body>
<div class="container centered-form">
    <div class="form-container">
        <h3 class="text-center">Введите номер клиента</h3>
        <form th:action="@{/check-phone}" method="post" class="needs-validation" novalidate>
            <div class="form-group">
                <input type="text" class="form-control" th:field="*{client.phoneNumber}" placeholder="Номер клиента"
                       required pattern="\d{11}">
                <div class="invalid-feedback">
                    Пожалуйста, введите корректный номер (11 цифр).
                </div>
            </div>
            <div class="text-center">
                <button type="submit" class="btn btn-primary">Поиск</button>
                <a th:if="${client.getPhoneNumber() != null}" th:href="@{/new-problem}" class="btn btn-primary" id="newProblemButton">Новая проблема</a>
            </div>
            <div th:if="${error}" class="alert alert-danger mt-3" role="alert">
                <span th:text="${error}"></span>
            </div>
        </form>
    </div>
```

Рисунок 9 – Код начальной страницы, часть 2

```
    <div id="problem-section" class="form-container">
        <h3 class="text-center">Список проблем клиента</h3>
        <div th:if="${client.getPhoneNumber() == null}" class="alert alert-warning" role="alert">
            Введите номер телефона, чтобы увидеть список проблем.
        </div>
        <div th:if="${(#lists.isEmpty(problems) || problems == null) && client.getPhoneNumber() != null}" class="alert alert-info" role="alert">
            Проблем не найдено.
        </div>
        <div th:each="problem : ${problems}" class="card problem-container">
            <div class="card-body">
                <h5 class="card-title" th:text="${problem.title}">Название проблемы</h5>
                <p class="card-text" th:text="${problem.resolution}">Решение проблемы</p>
                <small class="text-muted" th:text="${problem.getFormattedCreationDate()}">Дата создания</small>
            </div>
        </div>
    </div>
</div>
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.min.js"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
<script>
    // Bootstrap validation
    (function() {
        'use strict';
        window.addEventListener('load', function() {
            var forms = document.getElementsByClassName('needs-validation');
            Array.prototype.filter.call(forms, function(form) {
                form.addEventListener('submit', function(event) {
                    if (form.checkValidity() === false) {
                        event.preventDefault();
                        event.stopPropagation();
                    }
                    form.classList.add('was-validated');
                }, false);
            });
        }, false);
    })();
```

Рисунок 10 – Код начальной страницы, часть 3

```
    // Show problem section if client is present
    (function() {
        if (document.getElementById('problem-section')) {
            document.getElementById('problem-section').style.display = 'block';
        }
    })();
</script>
</body>
</html>
```

Рисунок 11 – Код начальной страницы, часть 4

После нажатия на кнопку «Новая проблема» открывается окно выбора новой проблемы клиента показанная на рисунке 12. HTML код этого окна показан на рисунках 13 – 14. Данное окно состоит из следующих частей:

– заголовок страницы: отображает название страницы и основное действие;

– список доступных проблем: содержит список проблем, которые могут быть выбраны оператором;
– форма для создания проблемы: скрытая форма, автоматически заполняемая и отправляемая при выборе проблемы.

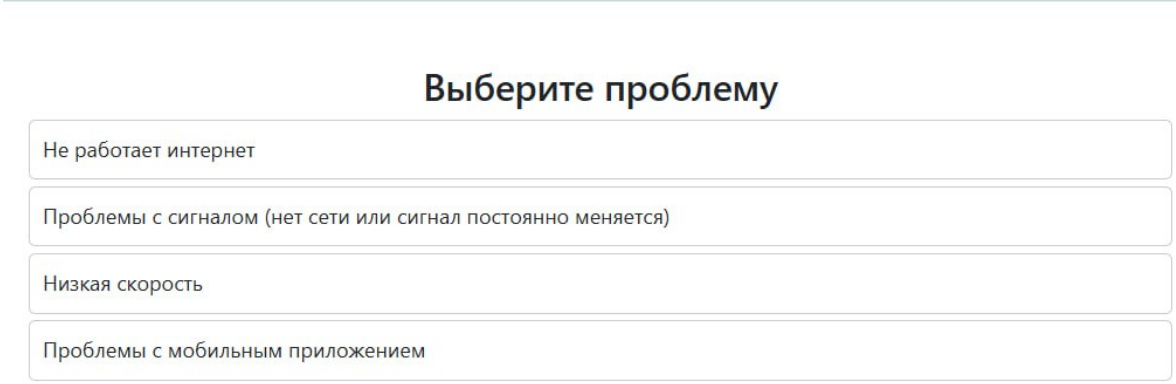


Рисунок 12 – Форма выбор проблемы

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Создание новой проблемы</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <style>
        .container {
            max-width: 900px;
        }
        .problem-list {
            list-style-type: none;
            padding: 0;
        }
        .problem-item {
            cursor: pointer;
            padding: 10px;
            border: 1px solid #ccc;
            margin-bottom: 5px;
            border-radius: 5px;
            transition: background-color 0.3s;
            display: flex;
            align-items: center;
        }
        .problem-item:hover {
            background-color: #f8f9fa;
        }
    </style>
</head>
<body>
<div class="container">
    <h3 class="text-center mt-5">Выберите проблему</h3>
    <ul class="problem-list">
```

Рисунок 13 – Код страницы выбора проблемы, часть 1

```
        <li th:each="problemType : ${problemTypeDescriptions}"
            class="problem-item"
            th:text="${problemType}"
            th:data-title="${problemType}"></li>
    </ul>
    <form th:action="@{/create-problem}" method="post" id="problemForm">
        <input type="hidden" name="problemTitle" id="problemTitle">
    </form>
</div>
<script>
    document.querySelectorAll('.problem-item').forEach(item => {
        item.addEventListener('click', function() {
            document.getElementById('problemTitle').value = this.dataset.title;
            document.getElementById('problemForm').submit();
        });
    });
</script>
</body>
</html>
```

Рисунок 14 – Код страницы выбора проблемы, часть 2

После выбора проблемы клиента, в новом окне открывается первый шаг процедуры. На рисунке 15 показан первый шаг процедуры «Не работает интернет».

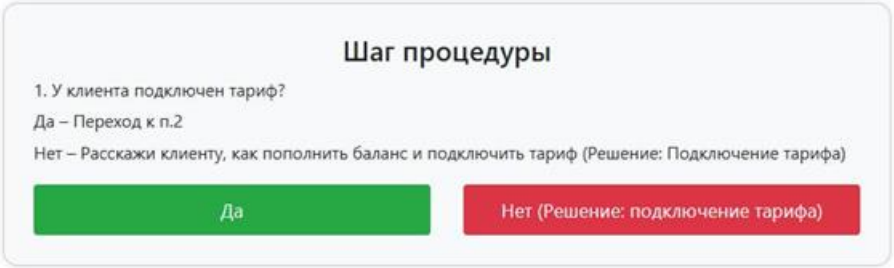


Рисунок 15 – Первый шаг процедуры «Не работает интернет»

HTML код первого шага процедуры показан на рисунках 16 – 19.

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Шаг процедуры</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <style>
        .centered-form {
            display: flex;
            justify-content: center;
            align-items: center;
            min-height: 100vh;
            flex-direction: column;
        }

        .form-container {
            border: 1px solid #ccc;
            border-radius: 15px;
            padding: 30px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            background-color: #f8f9fa;
            width: 100%;
            max-width: 900px;
        }

        .button-group {
            display: flex;
            justify-content: space-between;
            margin-top: 20px;
        }
```

Рисунок 16 – Код первого шага процедуры, часть 1

```
        .btn-custom {
            font-size: 1.2rem;
            padding: 10px 20px;
            width: 48%;
        }

        .text-body {
            font-size: 1.1rem;
            margin-top: 10px;
        }

        .text-body p {
            margin-bottom: 5px;
        }
    </style>
</head>
<body>
<div class="container centered-form">
    <div class="form-container">
        <h3 class="text-center">Шаг процедуры</h3>
        <div class="text-body">
            <p>1. У клиента подключен тариф?</p>
            <p>Да - Переход к п.2</p>
            <p>Нет - Расскажи клиенту, как пополнить баланс и подключить тариф (Решение: Подключение тарифа)</p>
        </div>
        <div class="button-group">
            <button id="yesButton" class="btn btn-success btn-custom">Да</button>
```

Рисунок 17 – Код первого шага процедуры, часть 2

```
            <button id="noButton" class="btn btn-danger btn-custom">Нет (Решение: подключение тарифа)</button>
        </div>
    </div>
</div>
<script>
    document.getElementById('yesButton').addEventListener('click', function () {
        window.location.href = '/signal-type-network-internet-check'; // Перенаправление на следующий шаг
    });

    document.getElementById('noButton').addEventListener('click', function () {
        saveProblem('Решение: подключение тарифа'); // Сохранение проблемы со статусом "Подключение тарифа"
    });

    function saveProblem(resolution) {
        const xhr = new XMLHttpRequest();
        xhr.open('POST', '/save-problem', true);
        xhr.setRequestHeader('Content-Type', 'application/json;charset=UTF-8');
        xhr.onreadystatechange = function () {
            if (xhr.readyState === 4 && xhr.status === 200) {
                alert('Проблема сохранена со статусом: ' + resolution);
                window.location.href = '/'; // Перенаправление на следующий шаг после сохранения
            }
        };
        const problem = {
            title: 'Проблема с тарифом',
            resolution: resolution
```

Рисунок 18 – Код первого шага процедуры, часть 3

```
        xhr.send(JSON.stringify(problem));
    }
</script>
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.min.js"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

Рисунок 19 – Код первого шага процедуры, часть 4

Выводы по главе.

В результате выполнения третьей главы была разобрана и выбрана для реализации клиент-серверная архитектура, и рассмотрены основные составляющие информационной системы. На стороне сервера был

реализован основной функционал, включающий обработку бизнес-логики, что обеспечивает корректное выполнение всех операций, связанных с управлением проблемами клиентов. Для хранения и обработки данных была спроектирована база данных, которая включает все необходимые сущности и их связи, что позволяет эффективно управлять данными и обеспечивать их целостность. С клиентской стороны был спроектирован и разработан многофункциональный и гибкий пользовательский интерфейс. Интерфейс позволяет операторам сотовой связи легко и быстро взаимодействовать с системой, отображая все необходимые данные о клиентах и их проблемах. В процессе разработки интерфейса были учтены все основные требования к удобству и эффективности работы пользователей, что способствует повышению производительности и снижению вероятности ошибок при обработке запросов клиентов.

Глава 4 Тестирование разработанной информационной системы

4.1 Тестирование процедуры «Низкая скорость»

Шаг 1. Вписываем в форму для поиска телефона (рисунок 20) номер, который не записан в базе данных, например, 79198189264 и нажимаем кнопку «Поиск».

Ожидаемый результат: номер записывается в базу данных, открывается список проблем с информацией, где указано, что проблем не найдено.

Фактический результат: номер записался в базу данных и открылся список проблем с информацией, где указано, что проблем не найдено.

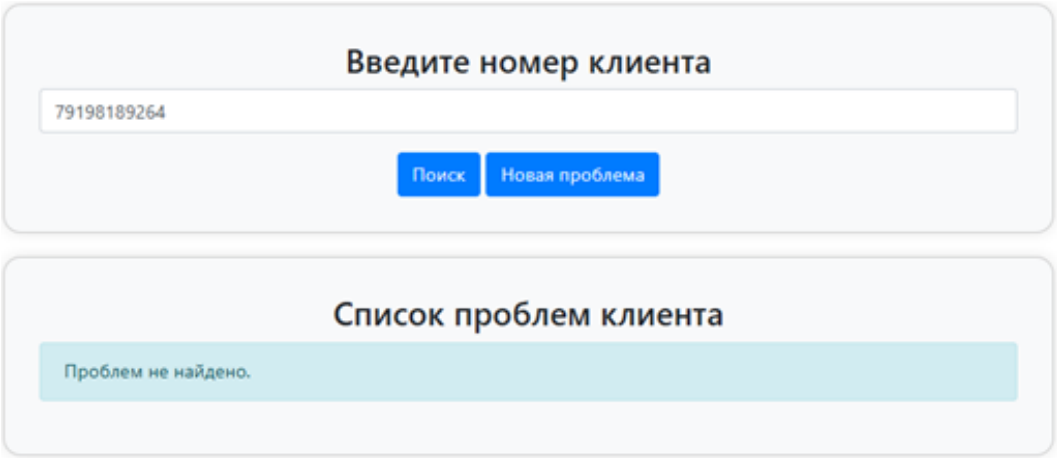


Рисунок 20 – Поиск номера в контрольном примере процедуры «Низкая скорость»

Шаг 2. Нажимаем кнопку «Новая проблема», которая показана на рисунке 20, после чего выбираем проблему клиента «Низкая скорость» в форме выбора проблемы (рисунок 21).

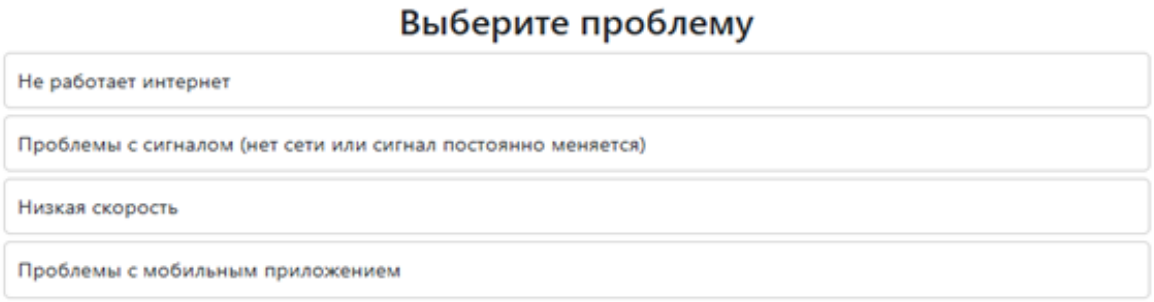


Рисунок 21 – Выбор проблемы «Низкая скорость»

Ожидаемый результат: открывается первый шаг процедуры «Низкая скорость» с вопросом «У клиента подключен тариф?».

Фактический результат: открылся первый шаг процедуры «Низкая скорость» с вопрос в окне «У клиента подключен тариф?».

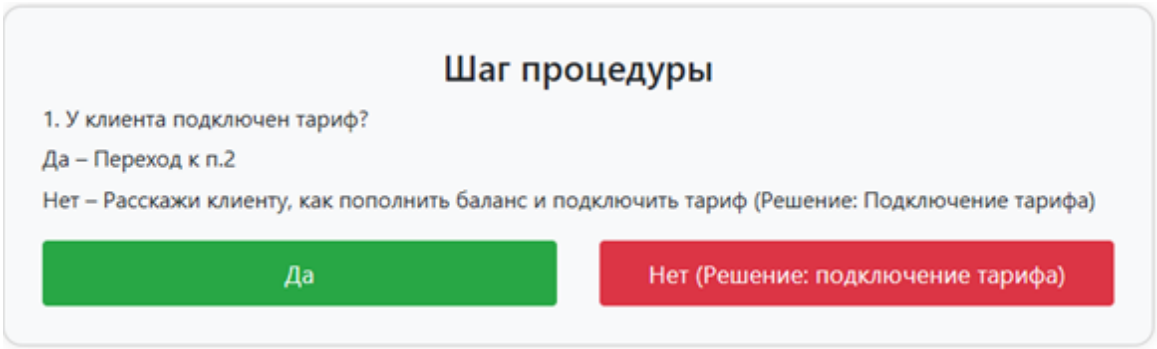


Рисунок 22 – Первый шаг процедуры «Низкая скорость»

Шаг 3. Нажимаем кнопку «Да», которая показана на рисунке 22.

Ожидаемый результат: переход на следующий шаг с вопросом «Сигнал и тип сети у клиента стабильный или меняется и падает меньше половины делений?».

Фактический результат: перешли на следующий шаг с вопросом «Сигнал и тип сети у клиента стабильный или меняется и падает меньше половины делений?».

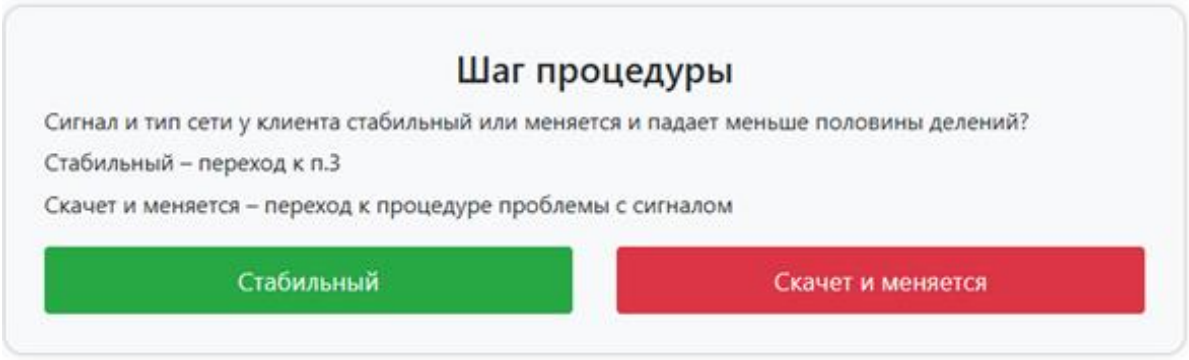


Рисунок 23 – Второй шаг процедуры «Низкая скорость»

Шаг 4. Нажимаем кнопку «Стабильный», которая показана на рисунке 23.

Ожидаемый результат: переход на следующий шаг с вопросом «Пользуется ли клиент VPN?».

Фактический результат: перешли на следующий шаг с вопросом «Пользуется ли клиент VPN?».

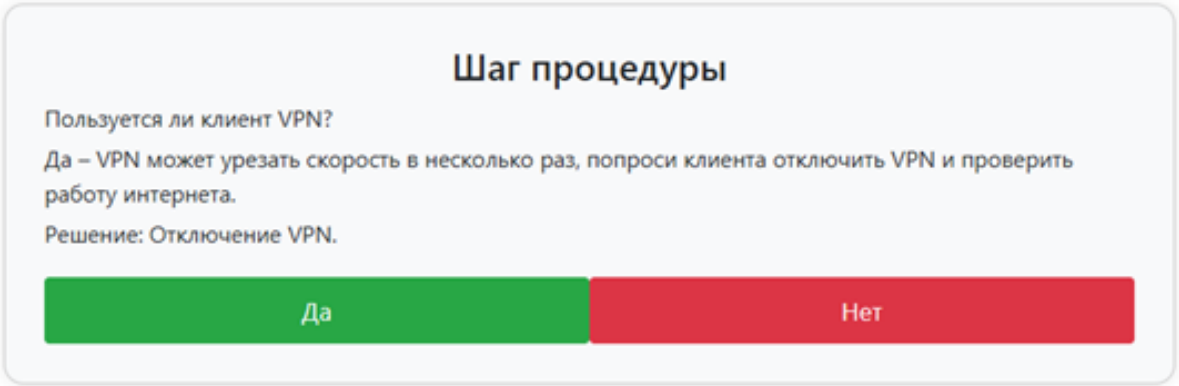


Рисунок 24 – Третий шаг процедуры «Низкая скорость»

Шаг 5. Нажимаем кнопку «Да», которая показана на рисунке 24.

Ожидаемый результат: выведется информационное сообщение: «Проблема сохранена со статусом: Отключение VPN».

Фактический результат: выведено информационное сообщение: «Проблема сохранена со статусом: Отключение VPN» (рисунок 25).

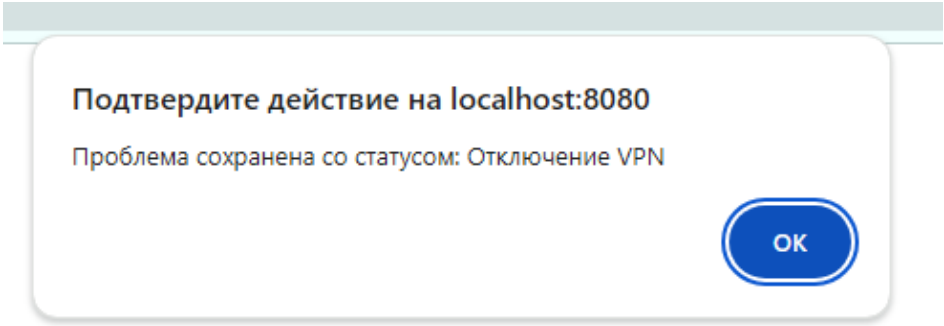


Рисунок 25 – Информационное сообщение «Проблема сохранена со статусом: Отключение VPN»

Шаг 6. Произвести повторный поиск номера 79198189264 на начальной странице.

Ожидаемый результат: в списке отображается проблема «Низкая скорость» с датой обращения клиента и решением «Отключение VPN».

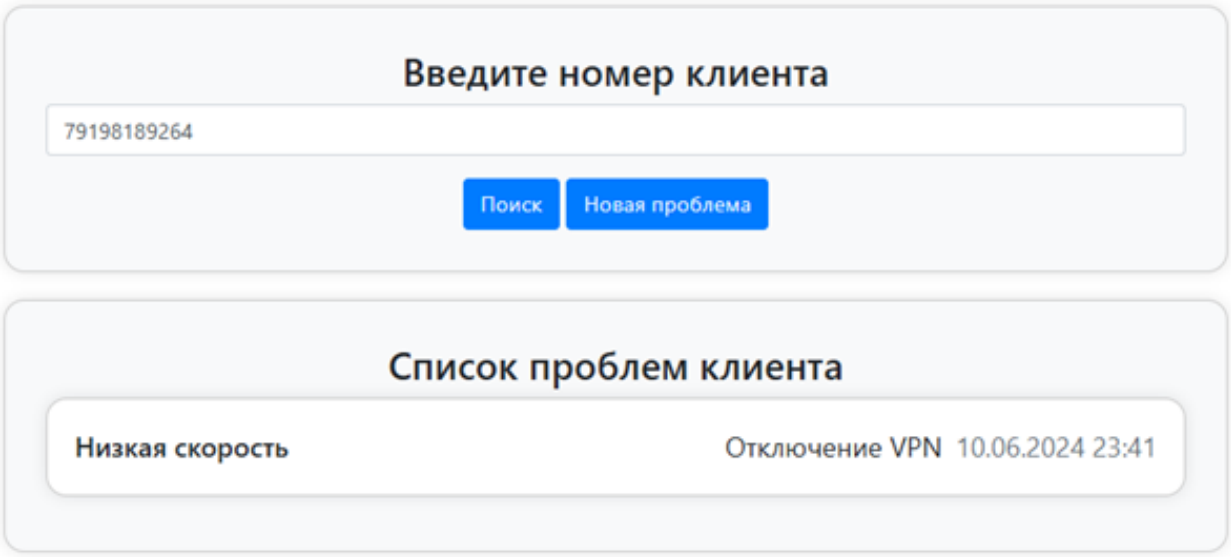


Рисунок 26 – Список проблем клиента после прохождения процедуры

Фактический результат: в списке (рисунок 26) отобразилась новая проблема «Низкая скорость» с датой 10.06.2024 23:41 и с решением «Отключение VPN».

4.2 Тестирование процедуры «Проблемы с мобильным приложением»

Шаг 1. Вписываем в форму для поиска телефона (рисунок 27) номер, который уже находится в базе данных, например, 79198189888, и нажимаем кнопку «Поиск».

Ожидаемый результат: в списке проблем отображаются проблемы клиента, если они есть.

Фактический результат: в списке проблем отображаются проблемы клиента.

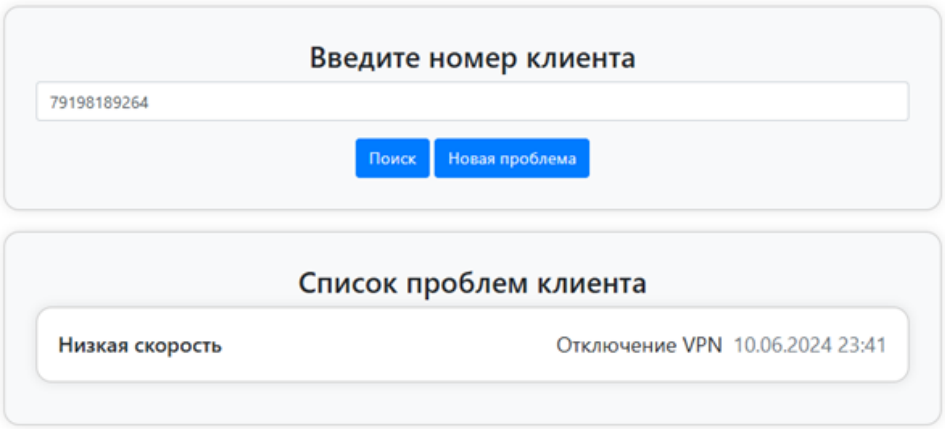


Рисунок 27 – Поиск номера, записанного в базу данных

Шаг 2. Нажимаем кнопку «Новая проблема», которая показана на рисунке 27, после чего выбираем проблему клиента «Проблемы с мобильным приложением» в форме выбора проблемы (рисунок 28).

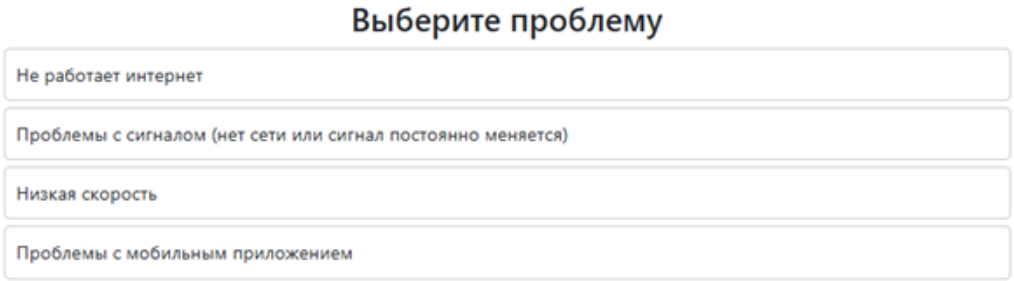


Рисунок 28 – Выбор проблемы «Проблемы с мобильным приложением»

Ожидаемый результат: открывается первый шаг процедуры «Проблемы с мобильным приложением» с информацией: «Проверь, нет ли сейчас активных аварий с мобильным приложением».

Фактический результат: открылся первый шаг процедуры «Проблемы с мобильным приложением» с информацией: «Проверь, нет ли сейчас активных аварий с мобильным приложением».

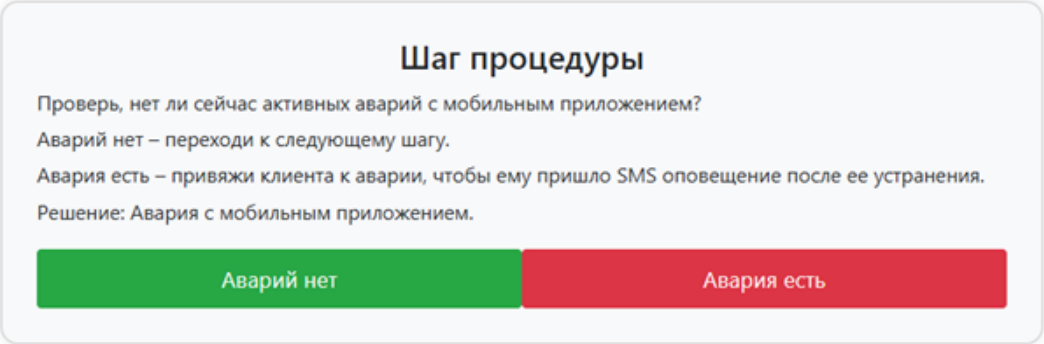


Рисунок 29 – Первый шаг процедуры «Проблемы с мобильным приложением»

Шаг 3. Нажимаем кнопку «Аварий нет», которая показана на рисунке 29.

Ожидаемый результат: открывается окно следующего шага с вопросом: «Уточни у клиента версию мобильного приложения».

Фактический результат: открылось окно с вопросом: «Уточни у клиента версию мобильного приложения».

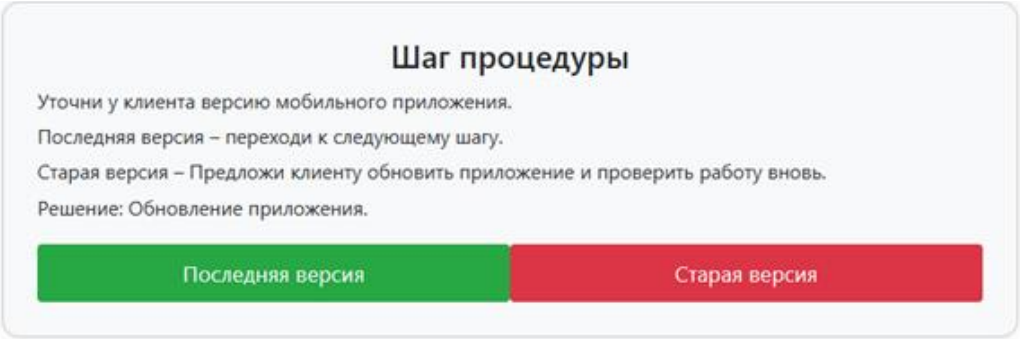


Рисунок 30 – Второй шаг процедуры «Проблемы с мобильным приложением»

Шаг 4. Нажимаем кнопку «Последняя версия», которая показана на рисунке 30.

Ожидаемый результат: должно открыться окно следующего шага с вопросом «Пользуется ли клиент VPN?».

Фактический результат: открылось окно с вопросом «Пользуется ли клиент VPN?».

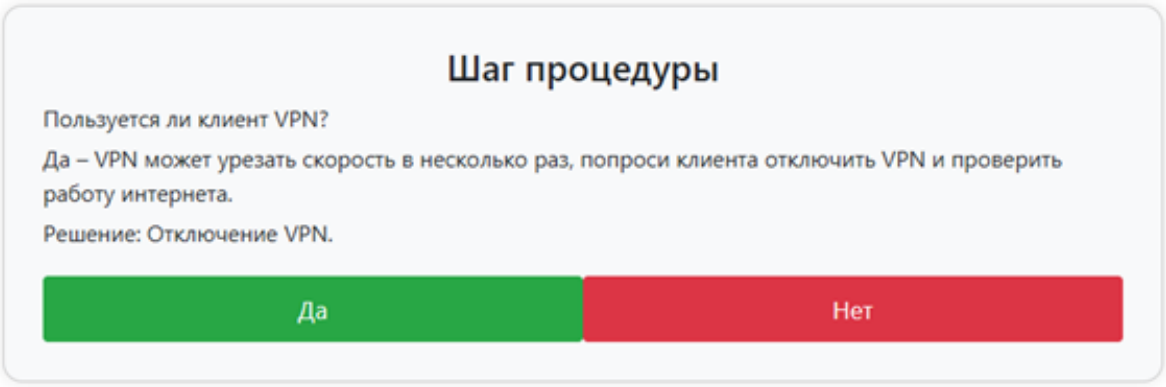


Рисунок 31 – Третий шаг процедуры «Проблемы с мобильным приложением»

Шаг 5. Нажимаем кнопку «Да», которая показана на рисунке 31.

Ожидаемый результат: выводится итоговое сообщение: «Передай заявку в технический отдел по проблеме с мобильным приложением. Для заявки потребуется версия мобильного устройства, модель устройства, версия ОС, примерная дата и время начало проблемы».

Фактический результат: выводится итоговое сообщение: «Передай заявку в технический отдел по проблеме с мобильным приложением. Для заявки потребуется версия мобильного устройства, модель устройства, версия ОС, примерная дата и время начало проблемы» (рисунок 32).

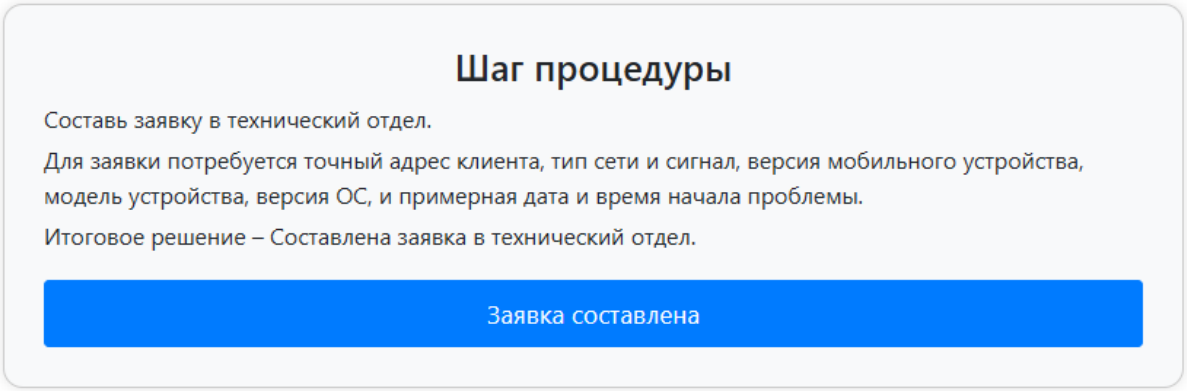


Рисунок 32 – Итоговое решение процедуры «Проблемы с мобильным приложением»

После нажатия кнопки «Заявка составлена» в базу данных записывается информация о предоставленном решение по проблеме.

4.3 Тестирование процедуры «Не работает интернет»

Шаг 1. Вписываем в форму для поиска телефона некорректный номер телефона (рисунок 33) и нажимаем кнопку «Поиск», например, 79198189.

Ожидаемый результат: выводится ошибка «Пожалуйста, введите корректный номер (11 цифр)».

Фактический результат: выводится ошибка «Пожалуйста, введите корректный номер (11 цифр)».

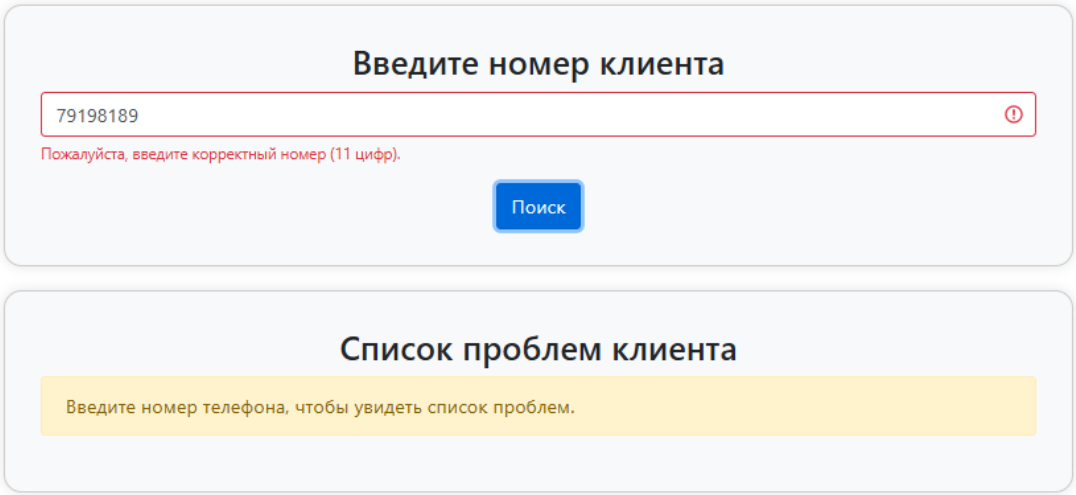


Рисунок 33 – Поиск некорректного номера

Шаг 2. Вписываем в форму для поиска телефона корректный номер телефона (рисунок 34) и нажимаем кнопку «Поиск», например, 79198189888.

Ожидаемый результат: в списке проблем отображаются проблемы клиента, если они есть.

Фактический результат: в списке проблем отображаются проблемы клиента.

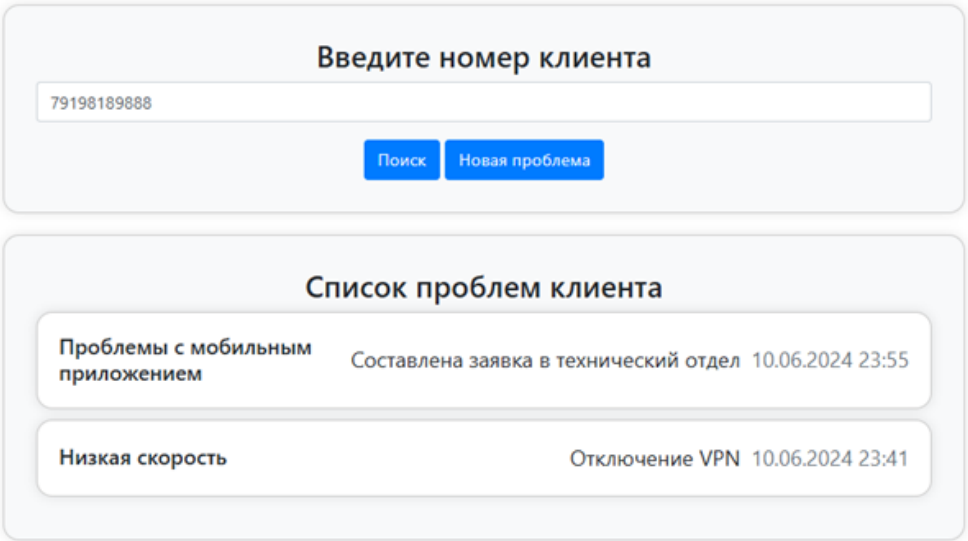


Рисунок 34 - Поиск корректного номера

Шаг 3. Нажимаем кнопку «Новая проблема», которая показана на рисунке 34, после чего выбираем проблему клиента «Не работает интернет» (рисунок 35).

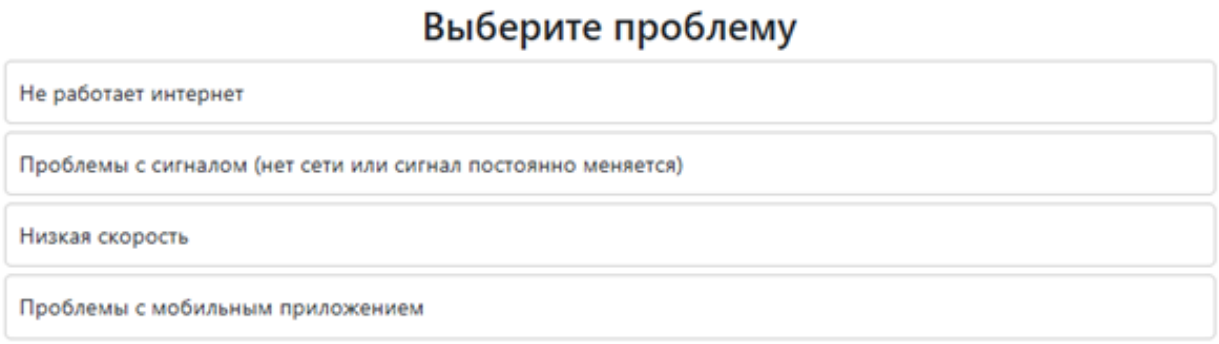


Рисунок 35 – Выбор проблемы «Не работает интернет»

Ожидаемый результат: открывается первый шаг процедуры «Не работает интернет» с вопросом «У клиента подключен тариф?».

Фактический результат: открылся первый шаг процедуры «Не работает интернет». Вопрос в окне «У клиента подключен тариф?».

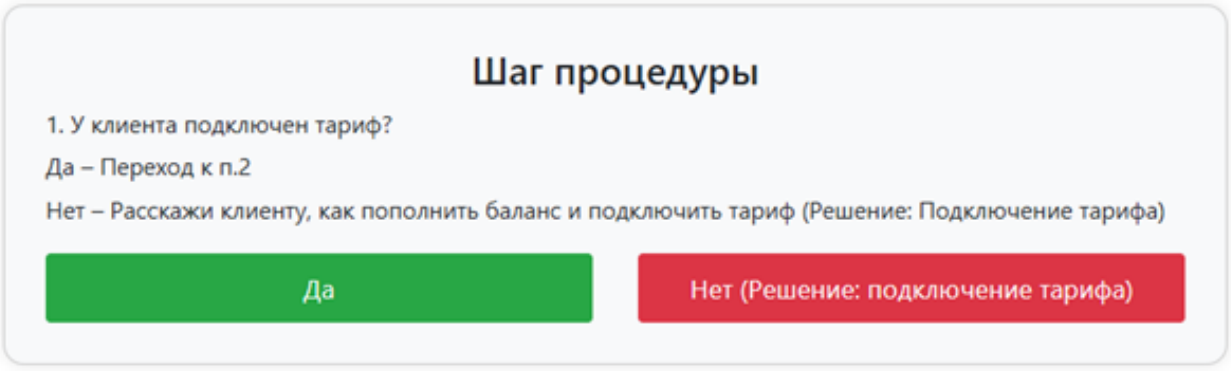


Рисунок 36 – Первый шаг процедуры «Не работает интернет»

Шаг 4. Нажимаем кнопку «Да», которая показана на рисунке 36.

Ожидаемый результат: переходим на следующий шаг с вопросом «Сигнал и тип сети у клиента стабильный или меняется и падает меньше половины делений?».

Фактический результат: перешли на следующий шаг с вопросом «Сигнал и тип сети у клиента стабильный или меняется и падает меньше половины делений?».

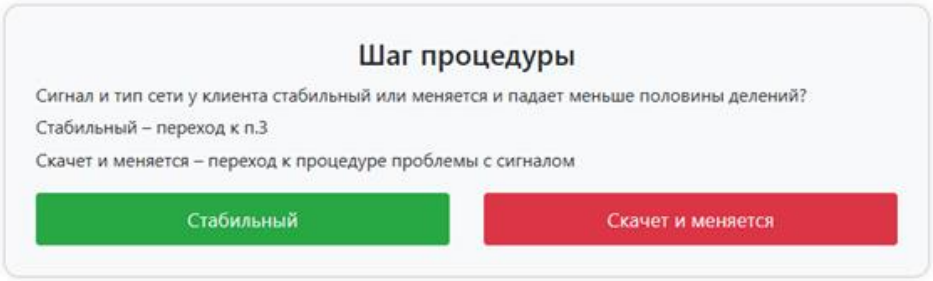


Рисунок 37 – Второй шаг процедуры «Не работает интернет»

Шаг 5. Нажимаем кнопку «Скачет и меняется», которая показана на рисунке 37.

Ожидаемый результат: запускается первый шаг процедуры «Проблемы с сигналом» с вопросом «Была за последние 7 дней зарегистрирована заявка по этой проблеме?»

Фактический результат: запустился первый шаг процедуры «Проблемы с сигналом» с вопросом «Была за последние 7 дней зарегистрирована заявка по этой проблеме?»

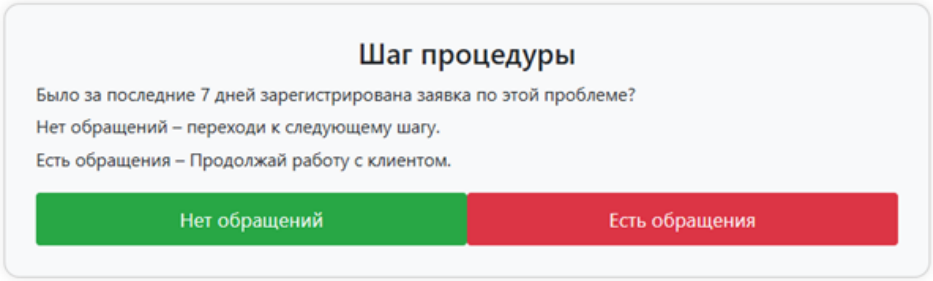


Рисунок 38 – Первый шаг процедуры «Проблемы с сигналом»

Шаг 6. Нажимаем кнопку «Нет обращений», которая показана на рисунке 38.

Ожидаемый результат: переходим на следующий шаг с вопросом «Сложность наблюдается по одному адресу или везде?».

Фактический результат: перешли на следующий шаг с вопросом «Сложность наблюдается по одному адресу или везде?».

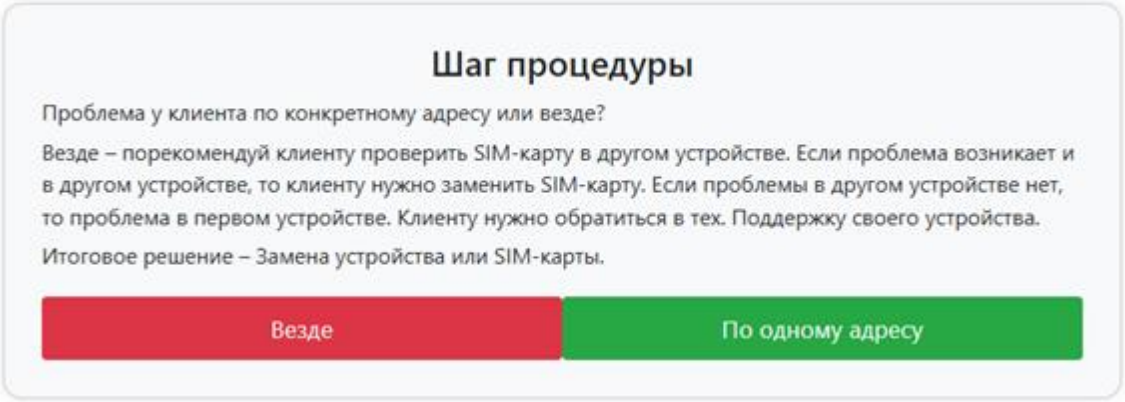


Рисунок 39 – Второй шаг процедуры «Проблемы с сигналом»

Шаг 7. Нажимаем кнопку «Везде», которая показана на рисунке 39.

Ожидаемый результат: выводится сообщение: «Порекомендуй клиенту проверить SIM-карту в другом устройстве. Если проблема возникает и в другом устройстве, то клиенту нужно заменить SIM-карту. Если проблемы в другом устройстве нет, то проблема в первом устройстве. Клиенту нужно обратиться в тех. поддержку своего устройства.»

Фактический результат: вывелось сообщение: «Порекомендуй клиенту проверить SIM-карту в другом устройстве. Если проблема возникает и в другом устройстве, то клиенту нужно заменить SIM-карту. Если проблемы в другом устройстве нет, то проблема в первом устройстве. Клиенту нужно обратиться в тех. поддержку своего устройства.»

Выводы по главе.

Подводя итоги тестирования информационной системы на основе пользовательских запросов, удалось доказать беспрецедентную работоспособность разработанного сервиса. В процессе тестирования были тщательно проверены все основные функции и возможности системы, начиная от ввода и обработки данных клиентов и проблем, до отображения результатов в пользовательском интерфейсе. В процессе проведения тестирования никаких ошибок не выявлено, было установлено, что сервис работает без существенных проблем и соответствует требованиям, предъявленным к нему, корректно реагируя на любые пользовательские запросы в контексте предметной области [5]. Таким образом, информационная система готова к внедрению и использованию в условиях реального рабочего процесса операторов сотовой связи.

Заключение

В ходе выполнения выпускной квалификационной работы была разработана информационная система поддержки принятия решений для операторов сотовой связи. Данная система направлена на улучшение качества и скорости обслуживания клиентов в колл-центре, что способствует повышению их лояльности и, следовательно, увеличению прибыли компании.

В разделе анализа предметной области был проведен обзор существующих систем поддержки принятия решений, выявлены их преимущества и недостатки, а также разработаны требования к создаваемой системе.

В разделе выбора средств разработки был произведен анализ популярных языков программирования и фреймворков. В результате были выбраны язык Java и фреймворк Spring для серверной части, а также шаблонизатор Thymeleaf для разработки интерфейса пользователя.

В разделе реализации системы была выполнена разработка серверной части приложения, включая создание базы данных на основе PostgreSQL для хранения информации о клиентах и их проблемах. Также был спроектирован и реализован удобный пользовательский интерфейс, который включает окна для ввода номера телефона клиента, отображения истории обращений и выбора новой проблемы.

В разделе тестирования была проведена проверка корректности работы системы. Для этого были разработаны и использованы контрольные примеры, которые позволили оценить эффективность работы приложения и убедиться в правильности реализованных алгоритмов.

В результате выполнения выпускной квалификационной работы была создана информационная система, которая помогает операторам принимать правильные решения быстро и точно. Все поставленные цели и задачи были

успешно достигнуты, а разработанная система готова к практическому применению.

Список используемых источников

1. Беннетт Дж. Wireless Nation: The Frenzied Launch of the Cellular Revolution in America. Издательство «Scribner», 2010. 432 с.

2. Блох Дж. Java. Эффективное программирование. Издательство «Вильямс», 2019. 456 с.

3. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Design Patterns: Elements of Reusable Object-Oriented Software. Издательство «Питер», 2019. 368 с.

4. Гетц Б. Java Concurrency in Practice. Издательство «Addison-Wesley», 2006. 384 с.

5. Горбачев Ю. Тестирование программного обеспечения. Базовый курс. ДМК Пресс, 2020. 464 с.

6. Дункан Р. Mobile Unleashed: The Origin and Evolution of ARM Processors in Our Devices. Издательство «Morgan & Claypool», 2017. 350 с.

7. Женко И., Наполи М. Pro Spring 5. Издательство «Apress», 2017. 680 с.

8. Карнеги Б. SQL Antipatterns: Avoiding the Pitfalls of Database Programming. Издательство «Pragmatic Bookshelf», 2010. 368 с.

9. Кеннеди Г. The Cellphone: The History and Technology of the Gadget That Changed the World. Издательство «Greenwood», 2016. 320 с.

10. Лонг Дж., Стадлер Ф. Thymeleaf in Action. Издательство «Manning Publications», 2017. 376 с.

11. Мангуэль А. A History of Reading. Издательство «Yale University Press», 1997. 388 с.

12. Мартин Р. Clean Code: A Handbook of Agile Software Craftsmanship. Издательство «Питер», 2017. 464 с.

13. Медведева Л. В., Рябова Е. А. Проектирование баз данных: Учебное пособие. Издательский центр «Академия», 2019. 384 с.

14. Михалча В. High Performance Java Persistence. Издательство «Amazon Digital Services», 2016. 412 с.

15. О'Бит Р., Майер С. PostgreSQL: Up and Running. Издательство «O'Reilly Media», 2017. 298 с.

16. Окс С. Java Performance: The Definitive Guide. Издательство «Орейли», 2018. 426 с.

17. Рамазанов Н. Проектирование интерфейсов: Лучшие практики и рекомендации. БХВ-Петербург, 2018. 256 с.

18. Уоллс К. Spring in Action. Издательство «Manning Publications», 2018. 520 с.

19. Хорстманн К. С. Java SE 8 for the Really Impatient. Издательство «Addison-Wesley», 2014. 240 с.

20. Шмидт М. Ф. The Book: A Global History. Издательство «Thames & Hudson», 2013. 352 с.