

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Волгоградский государственный технический университет»

Факультет Электроники и вычислительной техники

Кафедра Программное обеспечение автоматизированных систем

Согласовано

Утверждаю
Зав. кафедрой

(должность гл. специалиста предприятия)

О.А. Сычев

(подпись)

(инициалы, фамилия)

(подпись)

(инициалы, фамилия)

« ____ » _____ 20 ____ г.

« ____ » _____ 20 ____ г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к выпускной квалификационной работе бакалавра на тему
(наименование вида работы)

Создание веб-сервиса для индивидуального подбора и сравнения тарифов
мобильных операторов с учетом финансовых предпочтений пользователя

Автор Козарез Максим Вячеславович
(подпись и дата подписания) (фамилия, имя, отчество)

Обозначение ВКРБ–09.03.04–10.19–07–24
(код документа)

Группа ПрИн-466
(шифр группы)

Направление 09.03.04 – Программная инженерия,
Разработка программно-информационных систем
(код и наименование направления, наименование программы (профиля))

Руководитель работы Гилка В.В.
(подпись и дата подписания) (инициалы и фамилия)

Консультанты по разделам:

(краткое наименование раздела) (подпись и дата подписания) (инициалы и фамилия)

(краткое наименование раздела) (подпись и дата подписания) (инициалы и фамилия)

Нормоконтролер: _____
(подпись и дата подписания) (инициалы и фамилия)

Волгоград 2024 г.

Кафедра Программное обеспечение автоматизированных систем

« » 20 Г.

[illegible]

Перечень графического материала

- 1) _____

- 2) _____

- 3) _____

- 4) _____

- 5) _____

- 6) _____

- 7) _____

- 8) _____

- 9) _____

- 10) _____

- 11) _____

- 12) _____

Руководитель работы (проекта)

(подпись и дата подписания)

Гилка В.В.

(инициалы и фамилия)

Консультанты по разделам:

(краткое наименование раздела)

(подпись и дата подписания)

(инициалы и фамилия)

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Волгоградский государственный технический университет»

Кафедра «Программное обеспечение автоматизированных систем»

УТВЕРЖДАЮ:

Зав. кафедрой ПОАС

_____ О.А. Сычев

«__» _____ 20__ г.

Создание веб-сервиса для индивидуального подбора и сравнения тарифов
мобильных операторов с учетом финансовых предпочтений пользователя

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

ВКРБ–09.03.04–10.19–07–24–81

Листов **65**

Руководитель работы

_____ Гилка В.В.

«__» _____ 20__ г.

Нормоконтролер

Исполнитель

студент группы ПриИ-467

_____ Кузнецова А.С.

_____ Козарез Максим Вячеславович

«__» _____ 20__ г.

«__» _____ 20__ г.

Волгоград, 2024 г.

Аннотация

Настоящий документ является пояснительной запиской к выпускной квалификационной работе бакалавра на тему: «Создание веб-сервиса для индивидуального подбора и сравнения тарифов мобильных операторов с учетом финансовых предпочтений пользователя».

В работе обосновывается актуальность выбранной темы, проводится
....

Документ включает в себя страниц -..., рисунков -, приложений -...

Ключевые слова:

Содержание

Введение	7
1 Анализ виртуальных туров российских и зарубежных вузов	9
1.1 Введение в исследование	Ошибка! Закладка не определена.
Выводы	26
3 Реализация виртуального тура	27
3.1 Требования к функциональным характеристикам	27
Выводы	45
4 Тестирование виртуального тура	46
4.1 Mind Map карта областей тестирования	46
Выводы	62
Заключение	63
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	65
Приложение А	66
Справка о результатах проверки выпускной квалификационной работы на наличие заимствований	66
Приложение Б	67
Техническое задание	67
Приложение В	68
Руководство системного программиста	68

Введение

В условиях глобализации и стремительного развития цифровых технологий, мобильная связь становится неотъемлемой частью жизни современного человека. Важность выбора оптимального тарифного плана мобильной связи обусловлена растущими потребностями пользователей в доступе к качественным и доступным телекоммуникационным услугам. Поиск оптимального тарифа становится еще более актуальным на фоне постоянно меняющегося цифрового ландшафта, где новые технологии и услуги появляются с невероятной скоростью. С одной стороны, многообразие тарифов и услуг может казаться преимуществом, предоставляя потребителям широкий выбор. С другой стороны, это же многообразие часто приводит к путанице и затрудняет процесс принятия решения. Пользователи могут чувствовать себя перегруженными информацией и техническими деталями, что, в конечном итоге, может привести к выбору неоптимального тарифа, не соответствующего их реальным потребностям и финансовым возможностям.

В этом контексте появляется необходимость в инструменте, который мог бы упростить процесс выбора, предоставляя пользователю четкую, консолидированную и персонализированную информацию. Разработка веб-сервиса для индивидуального подбора и сравнения тарифов мобильных операторов, отвечающего этим критериям, становится решением, способным удовлетворить текущие потребности рынка. Такой сервис не только поможет пользователям сделать осознанный и экономически выгодный выбор, но и повысит общую прозрачность рынка мобильной связи.

В условиях постоянно растущего спроса на мобильную связь и ее услуги, разработка такого веб-сервиса становится не просто актуальной задачей, но и необходимостью, открывающей новые горизонты для удовлетворения потребностей современных пользователей в удобном, доступном и персонализированном выборе телекоммуникационных услуг.

Целью работы является – упрощение процесса поиска и экономия времени пользователя при выборе оптимального тарифа мобильных операторов.

Задачи:

- произвести анализ предметной области;
- произвести обзор существующих аналогов и выявить их преимущества и недостатки;
- определить требования к разрабатываемому веб-сервису;
- произвести проектирование базы данных исходя из требований;
- разработка веб-сервиса;
- протестировать разработанный веб-сервис и доказать его работоспособности и эффективность.

Объектом исследования в работе является процесс подбора и сравнения тарифов мобильных операторов.

Предметом исследования является процесс разработки и функционирования веб-сервиса для упрощения выбора и экономии времени при подборе тарифов мобильных операторов.

Методы исследований. Для решения поставленных задач были использованы методы математического моделирования, системного анализа, программной инженерии, объектно-ориентированного программирования, технологии проектирования человеко-машинного взаимодействия.

Разработанный веб-сервис значительно упрощает процесс выбора тарифов мобильной связи для конечных пользователей, предоставляя интуитивно понятный и легкий в использовании интерфейс. Пользователи могут быстро сравнивать различные тарифные планы и выбирать наиболее подходящие, что сокращает время, необходимое для принятия обоснованного решения. Подбор оптимального тарифа с учетом индивидуальных потребностей и финансовых возможностей пользователя помогает экономить средства, избегая переплат за ненужные услуги.

1 Анализ современного состояния рынка операторов мобильной связи

1.1 Характеристики проблемной области

Разработка веб-сервиса для изучения услуг, предоставляемых мобильными операторами, включает в себя анализ различных аспектов в области мобильного обслуживания и взаимодействия между потребителями и компаниями. Рассмотрим основные характеристики данной области исследования:

1. Потребительские Требования:

Современные пользователи активно стремятся к индивидуализированным услугам, даже в сфере тарифных планов мобильных операторов. Они ожидают, чтобы предлагаемые им тарифы соответствовали их уникальным потребностям и финансовым возможностям. Это включает в себя не только объем интернет-трафика, продолжительность звонков, но и дополнительные услуги, которые лучше соответствовали бы их повседневным активностям.

2. Сложность Выбора Тарифов:

С ростом количества мобильных операторов появляется множество тарифных планов, что создает путаницу для пользователя. Сложность выбора оптимального тарифа усиливается, учитывая разнообразие предложений. Пользователи часто сталкиваются с необходимостью учитывать финансовые ограничения, а также свои предпочтения при выборе тарифа.

3. Потребность в Эффективном Инструменте:

В условиях острой конкуренции мобильные операторы стремятся предложить эффективные инструменты для подбора тарифов, чтобы привлечь и удерживать клиентов. Создание веб-сервиса для индивидуального подбора

тарифов становится стратегически важным, открывая новые возможности для инноваций в области услуг связи и повышения удовлетворенности клиентов.

4. Роль Финансовых Предпочтений:

Финансовые возможности пользователя становятся ключевым фактором в принятии решения о выборе тарифного плана. Создание веб-сервиса, учитывающего бюджетные ограничения и финансовые предпочтения, придает дополнительную ценность для конечного пользователя, обеспечивая более адаптированные и экономически выгодные предложения.

5. Защита Персональных Данных:

С учетом чувствительности данных о финансах и личных предпочтениях, обеспечение высокого уровня защиты персональных данных становится приоритетом. Создание безопасного веб-сервиса требует внимательного подхода к защите информации, чтобы пользователи чувствовали себя уверенно при предоставлении своих данных.

Создание веб-сервиса для индивидуального подбора и сравнения тарифов мобильных операторов с учетом финансовых предпочтений пользователя представляет собой не только технологическое задание, но и стратегическую возможность повысить конкурентоспособность и улучшить взаимодействие с клиентами.

1.2 Описание существующих способов/процессов решения задачи

При описании существующих способов и процессов решения задачи следует учитывать различные подходы, которые могут быть использованы для

достижения поставленных целей. Рассмотрим существующие методы и практики, а также выявим их преимущества и недостатки.

1. Определение Целей и Задач:

Первым шагом в создании веб-сервиса является четкое определение целей проекта. Это может включать в себя улучшение пользовательского опыта, привлечение новых клиентов, повышение конкурентоспособности на рынке мобильной связи. Далее необходимо сформулировать конкретные задачи, такие как анализ рынка тарифов, разработка алгоритмов подбора и обеспечение безопасности данных.

Достоинства:

- Ясное направление: позволяет определить ясное направление для всего проекта.

Недостатки:

- Возможная неоднозначность: некоторые цели могут быть неоднозначными и требовать уточнений.
- Недостаточная конкретика: в некоторых случаях задачи могут быть сформулированы недостаточно конкретно, что затрудняет их выполнение.

2. Маркетинговый исследовательский анализ:

Проведение тщательного маркетингового исследования необходимо для полного понимания рынка мобильных операторов и тарифов. Важно выделить основные потребительские требования и предпочтения, проанализировать конкурентов и выявить их сильные и слабые стороны.

Достоинства:

- Полное понимание рынка: обеспечивает полное понимание рынка и конкурентной обстановки.
- Выявление требований: помогает выявить основные потребности и предпочтения потребителей и удовлетворить их в своем продукте.

Недостатки:

- Времязатратность: может требовать значительных временных и финансовых затрат.
- Недостаточная Объективность: Результаты могут быть подвержены субъективным оценкам и интерпретациям.

3. Определение Функциональности:

На этом этапе определяются основные функции веб-сервиса. Среди них - подбор тарифов, сравнение услуг, а также учет финансовых предпочтений пользователей. Также необходимо предусмотреть функции регистрации и авторизации для сохранения персональных настроек пользователей.

Достоинства:

- Четкость и Спецификация: обеспечивает четкость и спецификацию функциональных требований.
- Ориентированность на Пользователя: позволяет ориентироваться на потребности и предпочтения пользователей.

Недостатки:

- Ограничение на Начальном Этапе: некоторые функциональности могут быть ограничены или неудовлетворительны на начальном этапе.
- Изменение Требований: в случае изменения требований пользователей, может потребоваться пересмотр функциональности.

4. Разработка алгоритмов подбора:

Разработка эффективных алгоритмов подбора тарифов является одним из возможных этапов создания веб-приложения. Эти алгоритмы должны учитывать потребности пользователей, анализировать их финансовые предпочтения и предоставлять персонализированные рекомендации.

Достоинства:

- Персонализированные рекомендации: обеспечивает создание алгоритмов, способных предоставлять персонализированные рекомендации для каждого пользователя.
- Эффективность подбора: увеличивает эффективность потраченного пользователем времени на выбор подходящего тарифного плана.

Недостатки:

- Сложность разработки: требует высокой экспертизы в области алгоритмов и машинного обучения.
- Необходимость постоянного обновления: требует постоянного обновления и улучшения для соответствия изменяющимся требованиям.

5. Интерфейс и Дизайн:

Создание удобного интерфейса и дизайна веб-сервиса - важный этап. Интуитивная навигация, понятная структура страниц и привлекательный дизайн способствуют легкому взаимодействию пользователя с сервисом.

Достоинства:

- Привлекательный внешний вид: обеспечивает пользователю приятный опыт работы с сервисом и увеличивает шанс его повторного возвращения.

- Эффективность подбора: увеличивает эффективность потраченного пользователем времени на выбор подходящего тарифного плана.

Недостатки:

- Индивидуальные предпочтения: дизайн может не всегда соответствовать индивидуальным предпочтениям различных пользователей.
- Ограничения на устройствах: некоторые дизайнерские решения могут иметь ограничения на определенных устройствах или браузерах.

6. Разработка и Тестирование:

После определения функциональности начинается разработка веб-сервиса с использованием современных технологий. Важным этапом является тщательное тестирование функциональности и безопасности для выявления и устранения ошибок.

Достоинства:

- Гарантия функциональности: тестирование обеспечивает гарантию функциональности и безопасности веб-сервиса.
- Выявление ошибок: позволяет выявить и устранить ошибки на ранних этапах разработки.

Недостатки:

- Вреязатратность: тестирование может потребовать значительных временных затрат.
- Невозможность полного покрытия: невозможность покрытия всех возможных сценариев использования допускает возможность возникновения ошибок.

7. Интеграция Систем Безопасности:

Обеспечение защиты данных пользователя при их сборе и обработке - критически важный момент. На этом этапе веб-сервис интегрируется с системами безопасности, включая шифрование и меры защиты от несанкционированного доступа.

Достоинства:

- Защита персональных данных: гарантирует высокий уровень защиты персональных данных пользователей от несанкционированного доступа.
- Соблюдение нормативов: помогает соблюдать законодательные нормы и требования по обработке и хранению конфиденциальных данных.

Недостатки:

- Затраты на разработку и интеграцию: внедрение систем безопасности может потребовать дополнительных затрат на разработку и интеграцию.
- Возможное замедление работы сервиса: некоторые меры безопасности, такие как шифрование, могут вызвать замедление работы веб-сервиса.

1.3 Обзор существующих веб-сервисов, реализующих аналогичные функции.

1.3.1 Критерии анализа веб-сервисов

1. Интуитивность интерфейса

Легкость в понимании и использовании интерфейса для комфортного взаимодействия с сервисом.

2. Персонализированный подбор

Возможность получения рекомендаций, учитывающих индивидуальные потребности и предпочтения пользователя.

3. Сравнение тарифов и услуг

Предоставление пользователю ясной информации для сравнения различных тарифных планов и услуг.

4. Учет финансовых предпочтений

Характеристика: Возможность настройки параметров подбора в соответствии с финансовыми ограничениями пользователя.

5. Доступность на различных устройствах

Возможность использования сервиса на различных устройствах, обеспечивая удобство пользования.

6. Широкий выбор операторов

Возможность выбора тарифа среди широкого списка мобильных операторов (от 7 операторов мобильной связи).

1.3.2 Таблица с анализом аналогов

Критерий сравнения	Banki. ru	Срав ни	Unli m Tarif fs	Tarif er	Сравни.Та риф	Pro.traif.i nfo
Интуитивность интерфейса	-	-	+	+	-	-
Персонализирова нный подбор	-	-	+	+	+	+
Сравнение тарифов и услуг	+	+	+	+	+	-
Учет финансовых предпочтений	-	-	-	-	-	-
Доступность на различных устройствах	-	+	+	-	+	-

Широкий выбор операторов	-	+	-	-	+	+
--------------------------	---	---	---	---	---	---

2 Предлагаемый процесс подбора и сравнения тарифов мобильных операторов

2.1 Решение проблемы выбора тарифа мобильного оператора

Как ранее отмечалось, уже существует большое количество сайтов для подбора персонализированного тарифа. С помощью этих веб-сайтов можно не только быстро, но и удобно, не выходя из дома, выбрать тариф, подходящий под нужды конкретного пользователя. Однако, исходя из личного опыта и анализа существующих решений на рынке, пока что никому не удалось решить стоящую проблему полностью.

Для решения данной проблемы необходимо разработать комплексный веб-сервис. Этот сервис должен выполнять несколько ключевых функций, которые позволят пользователям быстро и удобно находить подходящие под их нужды тарифы.

Прежде всего, веб-сервис должен иметь возможность автоматически получать актуальную информацию о тарифах от различных мобильных операторов. Это можно реализовать с помощью интеграции с API операторов связи, либо с помощью парсинга веб-страниц, содержащих актуальные тарифные планы.

Хранение данных будут осуществляться в реляционной базе данных, которая обеспечит быстрый доступ и обновление информации. Наличие базы данных также поможет избежать потерю данных в случае аварийного завершения программы.

Предоставление пользователям возможности фильтрации тарифов, исходя из собственных предпочтениях – еще одна критически важная часть системы. Пользователи смогут указать свои предпочтения, такие как объем интернет-трафика, количество минут разговора, SMS. Также они смогут

задать финансовые ограничения. Эти данные будут использованы для анализа и подбора наиболее подходящих тарифов.

Сам веб-сервис будет состоять из фронтенд и бэкенд-частей, каждая из которых играет важную роль в обеспечении функциональности системы.

Фронтенд-часть веб-сервиса отвечает за взаимодействие с пользователем. Она будет предоставлять удобный и интуитивно понятный интерфейс, позволяющий пользователям вводить свои предпочтения, просматривать список доступных тарифов и переходить на страницу провайдера услуги. Интерфейс будет адаптирован для различных устройств, обеспечивая комфортное использование как на настольных компьютерах, так и на мобильных устройствах. Основные задачи фронтенд-части включают обработку пользовательских данных, отображение информации о тарифах и взаимодействие с бэкендом для получения актуальной информации.

Бэкенд-часть веб-сервиса будет обрабатывать запросы, поступающие от фронтенда, выполнять необходимую бизнес-логику и взаимодействовать с базой данных. Она будет отвечать за сбор и хранение данных о тарифах мобильных операторов, анализ пользовательских предпочтений и подбор оптимальных тарифных планов. Кроме того, бэкенд будет обеспечивать безопасность данных пользователей и управлять процессами регистрации и аутентификации. Для эффективного выполнения этих задач бэкенд должен быть хорошо структурирован и оптимизирован для обработки большого объема данных и запросов.

2.2 Формальная модель проблемной области

Формальная модель проблемной области включает в себя описание основных сущностей и их взаимодействий в контексте веб-сервиса для подбора и сравнения тарифов мобильных операторов. Основными

компонентами данной модели являются пользователи, мобильные операторы, тарифные планы и критерии выбора тарифов.

Пользователи — это конечные потребители веб-сервиса, которые ищут наиболее подходящие тарифные планы на основе своих финансовых предпочтений и потребностей в услугах мобильной связи. Каждый пользователь может иметь уникальные требования, такие как предпочтительная цена, количество минут для звонков, объем интернет-трафика и количество SMS.

Мобильные операторы — это компании, предоставляющие услуги мобильной связи. Каждый оператор имеет набор тарифных планов, которые различаются по стоимости и набору предоставляемых услуг.

Тарифные планы — это конкретные предложения мобильных операторов, включающие в себя набор услуг (минуты, интернет-трафик, SMS) и стоимость. Тарифные планы могут быть фиксированными или гибкими, позволяя пользователям выбирать дополнительные опции.

Критерии выбора тарифов — это параметры, по которым пользователи выбирают наиболее подходящий тарифный план. К ним относятся цена, объем интернет-трафика, количество минут и SMS, а также дополнительные услуги (например, безлимитные звонки внутри сети, роуминг и т.д.).

Взаимодействие между этими сущностями осуществляется следующим образом:

- пользователь вводит свои предпочтения и критерии выбора в веб-сервисе;
- веб-сервис собирает информацию о доступных тарифных планах от различных мобильных операторов;
- система сравнивает тарифные планы на основе введенных пользователем критериев;
- пользователь получает список наиболее подходящих тарифных планов;

2.3. Постановка задач на модели

Постановка задач на модели является важным этапом в разработке веб-сервиса для индивидуального подбора и сравнения тарифов мобильных операторов, поскольку она позволяет систематизировать и формализовать требования, определить основные цели и задачи, а также разработать алгоритмы для их решения.

Основная цель данного веб-сервиса состоит в том, чтобы предоставить пользователям возможность легко и эффективно подбирать и сравнивать тарифные планы мобильных операторов на основе их индивидуальных предпочтений и финансовых возможностей. Для достижения этой цели необходимо решить несколько ключевых задач.

Первая задача заключается в анализе и обработке данных о тарифных планах мобильных операторов. Это включает в себя сбор актуальной информации о тарифах, их характеристиках и условиях использования. Данные должны быть структурированы и организованы таким образом, чтобы их можно было легко анализировать и сравнивать. Важным аспектом является разработка базы данных, которая будет хранить информацию о тарифах и обеспечивать быстрый доступ к этим данным.

Вторая задача связана с разработкой пользовательского интерфейса, который будет интуитивно понятным и удобным для использования. Пользовательский интерфейс должен позволять вводить предпочтения и параметры поиска, такие как объем интернет-трафика, количество минут для звонков и количество SMS. Также интерфейс должен обеспечивать наглядное отображение результатов сравнения тарифных планов.

Третья задача включает тестирование и оптимизацию системы. Все компоненты веб-сервиса должны быть тщательно протестированы на функциональность, производительность и безопасность. Важно провести тестирование на устойчивость к внешним воздействиям и оптимизировать

производительность сервиса для обеспечения быстрого отклика на запросы пользователей.

Четвертая задача состоит в моделировании процессов, которые будут автоматизированы в рамках веб-сервиса. Это включает определение основных сценариев использования, разработку диаграмм, описывающих взаимодействие пользователя с системой, и определение ключевых процессов и их взаимосвязей. Построение модели процессов позволяет более точно определить требования к системе и разработать эффективные алгоритмы для решения поставленных задач.

Наконец, пятая задача связана с интеграцией и развертыванием системы. Необходимо определить архитектуру системы, выбрать технологии для реализации и разработать план интеграции всех компонентов. Также важно подготовить серверную инфраструктуру для развертывания веб-сервиса и обеспечить его масштабируемость для поддержки растущего числа пользователей.

Постановка задач на модели позволяет четко определить направление разработки веб-сервиса, установить приоритеты и структурировать процесс реализации, что является необходимым условием для успешного завершения проекта и достижения его целей.

2.4. Алгоритмы решения задач

Для того чтобы достичь поставленных целей – необходимо разработать алгоритм. Алгоритмы позволяют автоматизировать процесс обработки данных, анализа, а также сравнения тарифов для пользователей. Ниже приведены основные алгоритмы, разработанные для решения поставленных задач.

1. Алгоритм сбора и обновления данных о тарифных планах:

Первым шагом является создание алгоритма, который автоматически собирает и обновляет информацию о тарифных планах мобильных операторов. Этот алгоритм включает следующие этапы:

- Парсинг данных: использование веб-скрепинга для извлечения данных с официальных сайтов мобильных операторов и других надежных источников. Алгоритм должен уметь идентифицировать и извлекать ключевые параметры тарифных планов, такие как название, стоимость, объем интернет-трафика, количество минут и SMS.
- Очистка и нормализация данных: обработка собранных данных для удаления дубликатов, исправления ошибок и приведения информации к единому формату. Это необходимо для обеспечения корректного анализа и сравнения тарифов.
- Обновление базы данных: регулярное обновление информации в базе данных для обеспечения актуальности данных. Это может быть реализовано через периодические запросы к источникам данных или через API операторов, если они предоставляют такую возможность.

2. Алгоритм анализа и сравнения тарифных планов:

Этот алгоритм предназначен для анализа и сравнения тарифных планов на основе введенных пользователем предпочтений и параметров поиска. Основные шаги включают:

- Ввод данных пользователем: интерфейс позволяет пользователю вводить свои предпочтения, такие как желаемый объем интернет-трафика, количество минут для звонков, количество SMS и допустимая стоимость.
- Фильтрация тарифов: на основе введенных данных алгоритм отбирает тарифные планы, соответствующие заданным критериям. Это может быть

реализовано с помощью SQL-запросов к базе данных или использования специализированных библиотек для обработки данных.

- Сортировка и вывод результатов: тарифные планы сортируются по общему рейтингу, и пользователю отображаются наиболее подходящие варианты. Дополнительно может быть предусмотрена возможность сортировки по отдельным параметрам, например, по стоимости или объему интернет-трафика.

3. Алгоритм тестирования и оптимизации системы:

Для обеспечения высокой производительности и надежности веб-сервиса разработан алгоритм тестирования и оптимизации, который включает:

- Ручное тестирование: Проверка корректности работы всех функций веб-сервиса, включая ввод и обработку данных, генерацию рекомендаций и отображение результатов.
- Юзабилити тестирование: Тестирование, направленное на то, чтобы выяснить, насколько легко и удобно пользователю взаимодействовать с интерфейсом сайта.
- Нагрузочное тестирование: Тестирование, направленное на проверку способности программы эффективно работать при пиковых или очень высоких нагрузках, связанных с большим количеством запросов пользователей.

4. Алгоритм моделирования процессов:

Для моделирования процессов, которые будут автоматизированы в рамках веб-сервиса, разработан алгоритм, включающий следующие шаги:

- Определение сценариев использования: Анализ основных сценариев взаимодействия пользователя с системой, таких как поиск тарифов, сравнение и получение рекомендаций.

- Разработка диаграмм: Создание диаграмм, описывающих взаимодействие пользователя с системой, чтобы визуализировать ключевые процессы и их взаимосвязи.
- Определение ключевых процессов: Определение основных процессов, которые необходимо автоматизировать, и их взаимосвязей для более точного определения требований к системе.

5. Алгоритм интеграции и развертывания системы:

Для успешной интеграции и развертывания системы разработан алгоритм, включающий следующие шаги:

- Определение архитектуры системы: Выбор подходящей архитектуры системы для обеспечения её масштабируемости и надежности.
- Выбор технологий: Определение технологий для реализации веб-сервиса, включая выбор фреймворков, баз данных и инструментов для разработки.
- Разработка плана интеграции: Создание плана интеграции всех компонентов системы и подготовка серверной инфраструктуры для развертывания веб-сервиса.
- Масштабируемость: Обеспечение масштабируемости системы для поддержки растущего числа пользователей и адаптации к изменяющимся требованиям.

Эти алгоритмы смогут обеспечить эффективную реализацию задач, поставленных в рамках разработки веб-сервиса для индивидуального подбора и сравнения тарифов мобильных операторов, что способствует достижению основной цели проекта и удовлетворению потребностей пользователей.

Выводы

В ходе работы была создана формальная модель проблемной области, которая четко определила структуру данных и основные взаимосвязи. Это позволило сформулировать и детализировать задачи, необходимые для реализации функционала сервиса, включая интеграцию с базами данных операторов и обеспечение актуальности информации.

Разработка алгоритмов для решения этих задач была важным этапом, который включал методы фильтрации и сравнения тарифов, а также механизмы обеспечения безопасности и актуализации данных. Реализованные алгоритмы позволили создать эффективный инструмент для пользователей, обеспечивая точность и релевантность предоставляемой информации.

Таким образом, разработанный веб-сервис продемонстрировал свою эффективность и практическую ценность. Он значительно упрощает процесс выбора тарифных планов, экономит время пользователей и предоставляет им актуальную информацию, повышая их удовлетворенность. Кроме того, сервис способствует усилению конкуренции среди мобильных операторов, что может привести к улучшению качества предоставляемых услуг и снижению цен на тарифы.

3 Реализация веб-сервиса для индивидуального подбора и сравнения тарифов мобильных операторов с учетом финансовых предпочтений пользователя

3.1 Требования к функциональным характеристикам

Веб-сервис для индивидуального подбора и сравнения тарифов мобильных операторов должен обеспечивать выполнение следующих функций:

- отображение подробной информации о каждом тарифном плане (оператор, название тарифа, цена, объем интернет-трафика, количество минут разговора и SMS);
- предоставление ссылки на сайт официального провайдера каждого тарифного плана;
- возможность ввода и редактирования критериев для подбора тарифов (цена, объем интернет-трафика, количество минут разговора и SMS);
- отображение списка тарифов, соответствующих заданным критериям;

3.2 Требования к нефункциональным характеристикам

Нефункциональные характеристики веб-сервиса для индивидуального подбора и сравнения тарифов мобильных операторов определяют качество работы системы, ее производительность, надежность и удобство использования. Основные требования к нефункциональным характеристикам включают следующие аспекты:

- веб-сервис должен обеспечивать быструю обработку запросов пользователей и мгновенное отображение результатов поиска и сравнения тарифов. Время отклика системы не должно превышать 2-3 секунд даже при высокой нагрузке;
- система должна быть способна масштабироваться для обработки увеличивающегося количества пользователей и данных без снижения производительности. Это требует архитектуры, поддерживающей горизонтальное и вертикальное масштабирование;
- веб-сервис должен обеспечивать высокий уровень надежности и быть устойчивым к сбоям. В случае непредвиденных обстоятельств система должна быстро восстанавливаться и продолжать работу без потери данных;
- интерфейс веб-сервиса должен быть интуитивно понятным и удобным для пользователей. Дизайн должен быть адаптивным, обеспечивая корректное отображение на различных устройствах, включая компьютеры, планшеты и смартфоны;
- веб-сервис должен быть совместим с различными веб-браузерами;

3.3 Диаграмма вариантов использования и сценарии работы с веб-сервисом

Диаграмма вариантов использования веб-сервиса для пользователей представлена на рисунке 1.

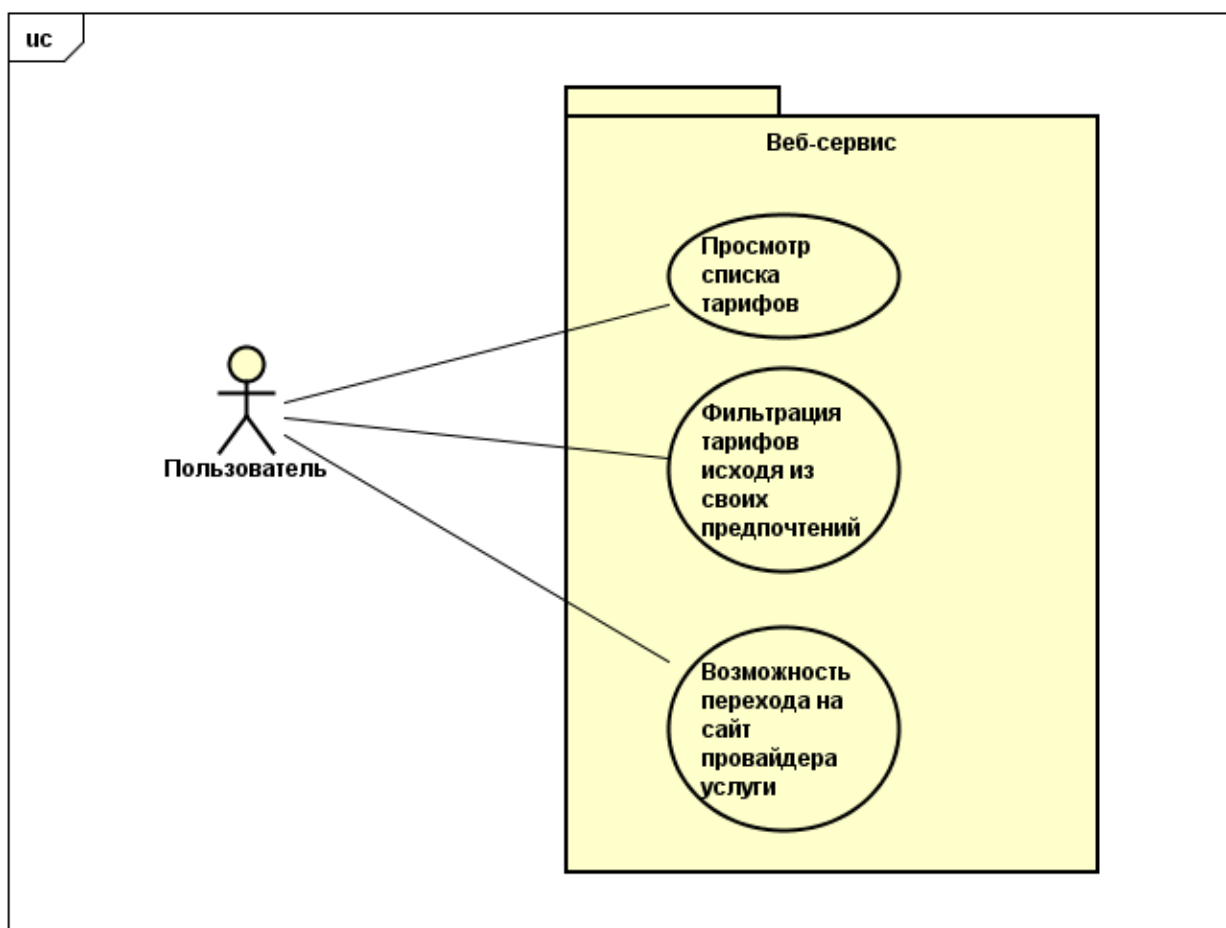


Рисунок 1 — Диаграмма вариантов использования

Сценарий «Просмотр списка тарифов мобильных операторов»:

- Пользователь заходит на главную страницу веб-сервиса
- Система отображает список всех доступных тарифов мобильных операторов

Сценарий «Фильтрация тарифов по параметрам»:

- Пользователь заходит на главную страницу веб-сервиса
- Вводит желаемые значения для фильтрации: цену, количество гигабайт, минут и сообщений
- Нажимает кнопку «Поиск»
- Система отображает отфильтрованный список тарифов, соответствующих заданным параметрам

Сценарий «Выбор интересующего тарифа»:

- Пользователь выбирает один из сценариев: «Просмотр списка тарифов мобильных операторов», «Фильтрация тарифов по параметрам»
- Пользователь нажимает на название интересующего тарифа
- Система переадресует пользователя на страницу провайдера услуги.

3.4 Общая структура веб-сервиса

Общая структура веб-сервиса для индивидуального подбора и сравнения тарифов мобильных операторов включает три ключевых компонента: клиентский интерфейс (Frontend), серверную часть (Backend) и базу данных. Эти компоненты взаимодействуют между собой, обеспечивая полноценное функционирование системы.

В данном проекте клиентский интерфейс разработан с использованием HTML и CSS, а также фреймворка Bootstrap, что обеспечивает простоту и интуитивность интерфейса для пользователей. Интерфейс позволяет пользователям вводить критерии поиска тарифов, такие как цена, объем интернет-трафика, количество минут для звонков и SMS. После ввода критериев и отправки запроса пользователи получают подробную информацию о каждом тарифном плане, включая название оператора, стоимость, объем трафика и другие важные характеристики. Также предоставляются ссылки на сайты официальных провайдеров для более детального ознакомления и подключения выбранного тарифа. Важным аспектом клиентского интерфейса является его адаптивность,

обеспечивающая корректное отображение информации на различных устройствах, таких как компьютеры, планшеты и смартфоны.

Серверная часть реализована на платформе языке Java с использованием фреймворка Spring, которая обеспечивает возможности для разработки масштабируемых приложений. Она отвечает за обработку всех запросов от клиентского интерфейса и выполнение бизнес-логики веб-сервиса. Когда пользователь вводит критерии поиска и отправляет запрос, серверная часть, используя Spring MVC, обрабатывает эти данные, выполняет поиск по базе данных и возвращает результаты на клиентский интерфейс. Серверная часть также управляет процессами аутентификации и авторизации пользователей, обеспечивая защиту данных и доступ к персонализированным функциям сервиса. Дополнительно, сервер интегрируется с внешними API мобильных операторов для получения актуальной информации о тарифах, что позволяет поддерживать данные в базе в актуальном состоянии.

База данных представляет собой реляционную базу данных, такую как PostgreSQL. Она хранит всю информацию о тарифных планах мобильных операторов, а также данные о пользователях и их предпочтениях. База данных структурирована и оптимизирована для быстрого выполнения запросов, что обеспечивает высокую производительность и минимальное время отклика на запросы пользователей. Информация о тарифах регулярно обновляется через интеграцию с внешними системами мобильных операторов. Это достигается за счет парсинга сайтов мобильных операторов, что позволяет поддерживать актуальность и достоверность предоставляемой информации.

Взаимодействие между этими компонентами осуществляется следующим образом: пользователь вводит критерии поиска на клиентском интерфейсе, данные отправляются на серверную часть посредством GET-запроса, где обрабатываются и выполняется поиск по базе данных. Результаты поиска возвращаются на клиентский интерфейс, где

отображаются в удобном формате. Такой подход обеспечивает быстрое и точное выполнение запросов, предоставляя пользователям актуальную и релевантную информацию о тарифных планах.

Таким образом, клиентский интерфейс, серверная часть и база данных вместе образуют единую систему, обеспечивающую эффективную работу веб-сервиса для подбора и сравнения тарифов мобильных операторов. Это взаимодействие позволяет пользователям легко и быстро находить подходящие тарифы, что значительно упрощает процесс выбора и повышает их удовлетворенность.

3.5 Проектирование Backend-части веб-сервиса

3.5.1 Проектирование схемы базы данных

Проектирование Backend-части приложения необходимо обычно начинается с проектирования схемы базы данных. На рисунке 2 представлена схема базы данных веб-сервиса для подбора и сравнения тарифов мобильных операторов:


rates	
id 	integer
provider_name	varchar
link	varchar
name	varchar
price	integer
messages	integer
minutes_of_call	integer
gigabytes_of_internet	integer

Рисунок 2 — Схема базы данных.

База данных содержит единственную таблицу rates, которая включает в себя следующие поля:

- id (тип: integer): уникальный идентификатор тарифного плана. Это поле является первичным ключом, обеспечивающим уникальность каждой записи в таблице.
- provider_name (тип: varchar): название мобильного оператора, предоставляющего тарифный план. Это поле хранит текстовые данные, представляющие название оператора.
- link (тип: varchar): ссылка на официальный сайт провайдера, где можно получить дополнительную информацию о тарифе или подключить его.
- name (тип: varchar): название тарифного плана. Это поле хранит текстовые данные, представляющие название тарифа.
- price (тип: integer): стоимость тарифного плана. Это поле хранит числовые данные, представляющие цену тарифа.
- messages (тип: integer): количество SMS, включенных в тарифный план. Это поле хранит числовые данные, представляющие количество сообщений.

- `minutes_of_call` (тип: `integer`): количество минут разговоров, включенных в тарифный план. Это поле хранит числовые данные, представляющие количество минут.
- `gigabytes_of_internet` (тип: `integer`): объем интернет-трафика, включенного в тарифный план. Это поле хранит числовые данные, представляющие объем трафика в гигабайтах.

Каждое из этих полей выполняет определенную роль в системе, обеспечивая хранение всех необходимых данных для сравнения и подбора тарифов мобильных операторов.

Проектирование схемы базы данных является важным этапом разработки Backend-части приложения, так как оно определяет структуру и организацию данных, обеспечивая их эффективное хранение и быстрый доступ. В данном случае схема базы данных спроектирована таким образом, чтобы обеспечивать хранение и управление информацией о тарифных планах, что позволяет легко искать и сравнивать различные тарифы по заданным критериям.

Схема базы данных является основой для реализации всей функциональности веб-сервиса. Она позволяет легко добавлять, обновлять и удалять информацию о тарифных планах, а также выполнять сложные запросы для поиска и сравнения тарифов по различным параметрам. Это обеспечивает высокую производительность и надежность системы, что является ключевыми требованиями для успешной работы веб-сервиса.

3.5.2 Проектирование backend архитектуры веб-сервиса

В основе веб-приложения для индивидуального подбора и сравнения тарифов мобильных операторов лежит архитектура REST API. Эта архитектура является устоявшимся стандартом для современных веб-

приложений благодаря своей гибкости и простоте. Это обеспечивает богатый выбор инструментов и решений.

Для разработки Backend-части веб-сервиса был выбран фреймворк Spring Boot. Это решение было принято по нескольким причинам. Во-первых, Spring Boot предоставляет модульную архитектуру, которая позволяет создавать структурированные приложения, использующие архитектуру MVC (Model-View-Controller). Это способствует лучшему разделению логики приложения и упрощает его поддержку и масштабирование. Во-вторых, Spring Boot обладает богатой экосистемой различных библиотек и инструментов, которые упрощают процесс разработки веб-сервиса. Если в самом Spring Boot отсутствует необходимая библиотека, можно легко найти и интегрировать сторонние библиотеки для Java.

Spring Boot также предоставляет встроенный сервер, что позволяет быстро и удобно разворачивать веб-сервис. Это особенно важно для тестирования и разработки, так как дает возможность мгновенно видеть изменения и тестировать функциональность в реальном времени.

Для управления зависимостями и сборки проекта используется Maven, что обеспечивает простоту и удобство управления проектом. Maven позволяет автоматически загружать и обновлять библиотеки, необходимые для проекта, а также упрощает процесс сборки и развертывания приложения. Благодаря Maven, разработка становится более организованной и эффективной, так как все зависимости и конфигурации управляются централизованно.

Таким образом, использование Spring Boot и Maven для разработки серверной части позволяет создавать надежное, масштабируемое и удобное в использовании веб-приложение, отвечающее современным стандартам и требованиям.

3.5.2.1 Взаимодействие веб-сервиса с базой данных

Взаимодействие веб-сервиса с базой данных осуществляется через DAO (Data Access Object) классы, которые интегрированы в архитектуру Spring Boot. Этот подход позволяет четко разделить бизнес-логику приложения от операций с базой данных, что улучшает читаемость и поддерживаемость кода.

Процесс взаимодействия веб-сервиса с базой данных можно описать следующим образом:

- модель данных: определяются классы сущностей (Entities), которые соответствуют таблицам базы данных. В нашем случае это класс Rate, который соответствует таблице rates и включает такие поля, как id, provider_name, link, name, price, messages, minutes_of_call и gigabytes_of_internet. Эти поля аннотируются с помощью JPA-аннотаций для указания их соответствия столбцам базы данных;
- DAO классы: для каждой сущности создается соответствующий DAO класс, который отвечает за выполнение операций с базой данных. Эти классы включают методы для выполнения операций CRUD (Create, Read, Update, Delete). Например, класс RateDAO будет содержать методы для добавления, обновления, удаления и поиска тарифных планов;
- конфигурация Spring: В конфигурации Spring определяется настройка для подключения к базе данных, а также создание и управление экземплярами DAO классов. Spring Boot автоматически конфигурирует подключение к базе данных с использованием параметров, указанных в файле настроек приложения application.properties;
- сервисный слой: в сервисном слое определяется бизнес-логика для работы с данными. Классы сервиса используют DAO классы для взаимодействия с базой данных. Например, класс RateService может содержать методы для поиска тарифов по заданным критериям, обновления информации о тарифах и удаления устаревших данных;

Контроллеры: В контроллерах реализуются конечные точки API, которые обрабатывают HTTP-запросы от клиентской части. Контроллеры используют сервисы для выполнения необходимой бизнес-логики и взаимодействия с базой данных. Например, RateController может содержать методы для получения списка тарифов по заданным критериям, добавления нового тарифа или обновления существующего.

Процесс работы веб-сервиса с базой данных можно проиллюстрировать следующим образом: когда пользователь отправляет запрос на получение списка тарифов, этот запрос попадает в контроллер, который вызывает соответствующий метод сервиса. Сервис, в свою очередь, использует DAO класс для выполнения запроса к базе данных, получает результаты, обрабатывает их и возвращает контроллеру. Контроллер формирует ответ и отправляет его обратно пользователю.

Использование DAO классов для работы с базой данных позволяет четко структурировать код, облегчает его тестирование и поддержку. Этот подход также способствует лучшему разделению ответственности между различными слоями приложения, что повышает его модульность и гибкость. Благодаря интеграции с Spring Boot, разработка и управление подключением к базе данных становятся простыми и удобными, что способствует быстрому и эффективному созданию надежного веб-сервиса.

3.5.2.2 Взаимодействие сервера с клиентским интерфейсом

Взаимодействие между сервером и клиентским интерфейсом обеспечивает обмен данными между пользователем и сервером, позволяя оперативно обновлять информацию на клиентской стороне без необходимости перезагрузки страницы. В данной секции рассматриваются используемые технологии, методы передачи данных и процесс обработки запросов.

Основой взаимодействия служат современные веб-технологии, такие как HTTP/HTTPS и RESTful API. HTTP/HTTPS используется для передачи данных по сети, причем HTTPS обеспечивает шифрование, что существенно повышает безопасность передачи данных. RESTful API предоставляет гибкий и расширяемый интерфейс для взаимодействия между клиентом и сервером, позволяя выполнять различные операции с данными.

Процесс передачи данных начинается с того, что клиентский интерфейс, при взаимодействии пользователя с веб-сервисом, отправляет запросы на сервер. Эти запросы могут включать различные типы операций, такие как получение данных о тарифах, добавление новых предпочтений пользователя или обновление существующих данных. Сервер, получив запрос, обрабатывает его с помощью внутренней логики и доступа к базе данных и затем начинает подготовку ответа в формате, понятном клиенту.

После обработки запроса сервер отправляет ответ обратно на клиентскую сторону. Клиентский интерфейс получает этот ответ и обновляет отображаемую информацию. Например, если пользователь ввел новые параметры поиска, клиентский интерфейс отправляет запрос на сервер, который возвращает соответствующие тарифные планы. Код на стороне клиента обновляет HTML-контент, отображая новые данные. Это делает работу с сервисом более быстрой и удобной для пользователя.

Этот подход позволяет пользователям получать актуальную информацию о тарифах мобильных операторов в реальном времени, делая работу с сервисом эффективной и приятной.

3.5.3 Проектирование парсера

Проектирование парсера, как и базы данных, является необходимым в рамках разработки веб-сервиса для подбора и сравнения тарифов мобильных операторов, поскольку он отвечает за сбор и обработку данных с веб-сайтов операторов. Для реализации парсера используется библиотека Jsoup, которая предоставляет мощные инструменты для работы с HTML и упрощает процесс извлечения данных из веб-страниц.

Jsoup - это Java-библиотека, которая позволяет эффективно парсить HTML и извлекать необходимую информацию. Прежде всего, необходимо подключить Jsoup к проекту, добавив соответствующую зависимость в файл сборки проекта `pom.xml` для Maven.

Затем необходимо определить все целевые страницы для парсинга. Иными словами, на этом этапе определяются веб-страницы мобильных операторов, с которых будет собираться информация о тарифных планах. Важно самостоятельно проанализировать HTML-структуру этих страниц, чтобы понять, где находятся необходимые данные и затем реализовать это в коде приложения.

Работа с Jsoup состоит из следующих шагов:

1. Загрузка HTML-страницы: сначала необходимо загрузить HTML-код страницы. Это можно сделать с помощью метода `Jsoup.connect(url).get()`, который загружает и парсит HTML из указанного URL
2. Извлечение нужных элементов: после загрузки HTML-страницы необходимо выбрать элементы, содержащие информацию о тарифах. Это можно сделать с помощью методов Jsoup для выбора элементов, таких как `select()`, которые используют CSS-подобные селекторы для поиска элементов.
3. Обработка и структурирование данных: Извлеченные данные нужно обработать и структурировать для дальнейшего использования. Например,

можно создать объекты тарифов и сохранить их в коллекции для последующего добавления в базу данных.

4. Необходимо предусмотреть обработку возможных ошибок и исключений, таких как проблемы с подключением, изменения в структуре HTML страниц или недоступность веб-сайтов. Это поможет сделать парсер более надежным и устойчивым к изменениям.
5. Для обеспечения актуальности данных процесс парсинга необходимо автоматизировать. Это можно сделать, настроив регулярное выполнение парсера с помощью встроенных средств Java.
6. В случае исправного извлечения тарифов с сайтов операторов, необходимо удалить из базы данных тарифы, оставшиеся с момента прошлого парсинга. Извлеченные и обработанные данные о тарифах сохраняются в базе данных для дальнейшего использования и анализа.

3.5.4 Диаграмма классов приложения

На рисунке 3 представлена диаграмма классов концептуального уровня Backend-части приложения.

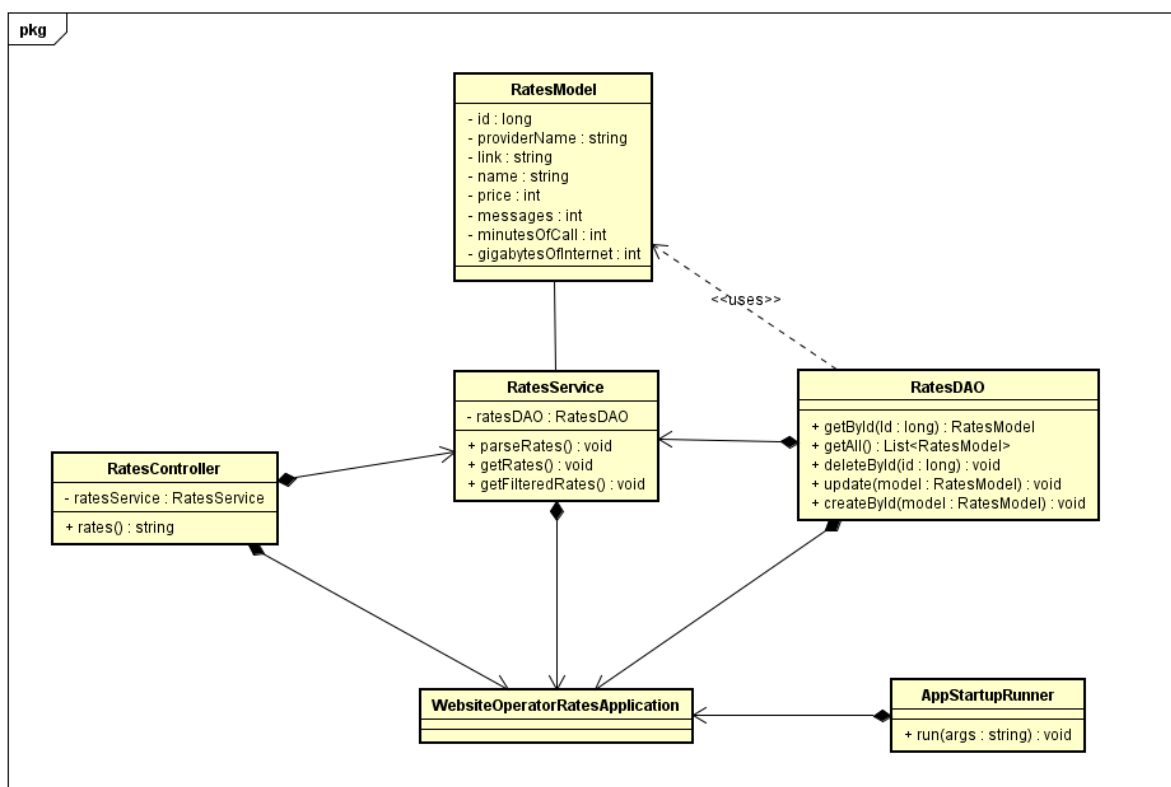


Рисунок 3 — Диаграмма классов концептуального уровня.

Диаграмма классов приложения иллюстрирует архитектуру и взаимодействие ключевых компонентов системы, предназначенной для выбора тарифов мобильных операторов. В этом приложении задействованы следующие основные классы: RatesModel, RatesDAO, RatesService, RatesController, WebsiteOperatorRatesApplication, и AppStartupRunner.

WebsiteOperatorRatesApplication представляет собой основной класс приложения, который управляет его жизненным циклом. Этот класс инициализирует все необходимые компоненты и запускает приложение, обеспечивая готовность всех сервисов к обработке запросов.

RatesModel представляет собой сущность, описывающую тарифный план. Этот класс инкапсулирует все необходимые данные о тарифе, включая имя провайдера, ссылку, название тарифа, стоимость, количество сообщений, минут звонков и гигабайт интернета. Модель тарифа используется для обмена данными между слоями приложения. RatesModel также отвечает за

представление данных в базе данных. Исходя из полей RatesModel происходит формирование схемы базы данных при старте приложения.

RatesDAO (Data Access Object) отвечает за доступ к данным тарифов. Этот класс предоставляет методы для получения, создания, обновления и удаления записей в базе данных. Он действует как посредник между базой данных и бизнес-логикой, обеспечивая абстракцию операций с данными.

RatesService реализует бизнес-логику приложения. Этот класс использует RatesDAO для взаимодействия с базой данных и предоставляет методы для обработки и фильтрации тарифных планов. Основная задача RatesService заключается в обработке данных и выполнении операций, необходимых для функционирования бизнес-логики.

RatesController является компонентом уровня представления в архитектуре MVC (Model-View-Controller). Он обрабатывает входящие HTTP-запросы от клиентов и использует RatesService для выполнения необходимых операций. RatesController отвечает за маршрутизацию запросов и возвращение ответов клиентам.

AppStartupRunner отвечает за выполнение начальной настройки и запуск приложения. Этот класс выполняет парсинг при старте приложения.

3.5.4.1 Взаимодействие классов

RatesController взаимодействует с RatesService для выполнения бизнес-логики в ответ на запросы клиентов.

RatesService использует RatesDAO для выполнения операций с данными, абстрагируя логику доступа к данным от бизнес-логики.

RatesDAO управляет доступом к базе данных и обеспечивает необходимые операции с данными тарифов.

WebsiteOperatorRatesApplication инициализирует AppStartupRunner, который выполняет парсинг при старте приложения, что позволяет всего держать состояние тарифов в актуальном состоянии.

Все компоненты взаимодействуют друг с другом через четко определенные интерфейсы и зависимости, что обеспечивает модульность, поддержку и расширяемость приложения.

Такое разделение обязанностей между классами способствует созданию структурированной, легко поддерживаемой и расширяемой архитектуры приложения, что позволяет эффективно решать поставленную задачу по выбору тарифов мобильных операторов.

3.6 Используемые программные средства

3.6.1 Программные средства, используемые при разработке Backend-части приложения

Backend-часть приложения, ответственная за обработку данных и взаимодействие с базой данных, построена с использованием Spring Boot. Этот фреймворк позволяет создавать приложения на Java с минимальной конфигурацией благодаря встроенным механизмам автоконфигурации и стартерам. Мы выбрали Java 17 для разработки, так как это последняя версия языка, предоставляющая новые возможности, и она имеет долгосрочную поддержку (LTS).

Управление зависимостями и сборка проекта осуществлялись при помощи Maven. Файл pom.xml содержит информацию о зависимостях и плагинах, необходимых для сборки проекта. Я использовал PostgreSQL в качестве реляционной базы данных для хранения информации о тарифах

мобильных операторов. Для взаимодействия с базой данных был использован Hibernate, который предоставляет удобный способ сопоставления объектов Java с записями в базе данных.

Spring ORM обеспечил интеграцию между Spring Framework и Hibernate, упрощая настройку и использование ORM в проекте. Для уменьшения количества шаблонного кода и улучшения читаемости кода была использована библиотека Lombok, которая способна заменить большой пласт примитивного и однообразного кода, такого как: геттеры, сеттеры и конструкторы одной аннотацией к классу. Jsoup был выбран для парсинга HTML страниц операторов связи и извлечения информации о тарифах.

Для сборки и упаковки Spring Boot приложения был использован плагин Spring Boot Maven Plugin, который позволяет легко управлять зависимостями и создавать исполняемые JAR-файлы. Сами зависимости были взяты с официального сайта – Maven repository.

3.6.2 Программные средства, используемые при разработке Frontend-части приложения

Как было сказано ранее, Frontend-часть приложения занимается отображением данных для пользователя. Для создания структуры веб-страниц был использован HTML, а CSS помог в стилизации и оформлении страниц. Для динамической генерации HTML контента на сервере был выбран шаблонизатор FreeMarker, который интегрирован с Backend-частью через Spring Boot.

Bootstrap был выбран как фреймворк для разработки адаптивных веб-интерфейсов, предоставляющий готовые компоненты CSS и JavaScript для ускорения процесса разработки и обеспечения единообразного стиля приложения. Это позволило создать интерфейс пользователя, который

отлично воспроизводится как на экранах компьютеров, так на мобильных устройствах и планшетах.

Выводы

В данной главе был описан процесс разработки веб-сервиса для индивидуального подбора и сравнения тарифов мобильных операторов с учетом финансовых предпочтений пользователя. Основные этапы реализации включают определение требований к функциональным и нефункциональным характеристикам системы, разработку архитектуры приложения, а также выбор и использование необходимых программных средств для создания как backend, так и frontend частей.

Были определены ключевые функции веб-сервиса, включая отображение подробной информации о тарифных планах, предоставление ссылок на сайты операторов, возможность ввода и редактирования критериев для подбора тарифов, а также отображение списка тарифов, соответствующих заданным критериям. По сему были рассмотрены функциональные и нефункциональные требования.

Помимо этого, была спроектирована архитектура приложения, основой для которой стал Spring Boot для backend и Freemarker совместно с Bootstrap для frontend. Это позволило создать структуру приложения, которая легко поддерживается и расширяется, а также обеспечивает эффективную работу с базой данных и отображение информации пользователям.

Диаграмма классов на концептуальном уровне показывает взаимодействие основных компонентов системы: контроллеры, сервисы и классы доступа к данным. Такое разделение обязанностей между классами способствует созданию структурированной архитектуры, что позволяет эффективно решать задачи по выбору тарифов мобильных операторов.

Также стоит отметить, что были во всех подробностях описаны программные средства, использованные для реализации как Backend-части приложения, включая базу данных и парсер, так и Frontend-части веб-сервиса.

4 Тестирование веб-сервиса для индивидуального подбора и сравнения тарифов мобильных операторов

Для проверки стабильной работы программы необходимо произвести тестирование веб-приложения. Как говорилось раньше, программа будет проходить 3 вида тестирования: ручное тестирование для проверки общей работоспособности приложения, юзабилити тестирование для оценки удобства работы с веб-приложением независимыми пользователями, а также нагрузочное тестирование, для проверки стабильной работы приложения в условиях большого количества одновременно взаимодействующих с приложением пользователей.

4.1 Ручное тестирование

Ручное тестирование играет важную роль в обеспечении качества программного обеспечения. Этот процесс включает в себя выполнение тестовых сценариев вручную, что позволяет проверить функциональность, пользовательский интерфейс и общее поведение приложения изнутри.

Первый шаг в ручном тестировании - это разработка тест-кейсов или тест-скриптов, которые описывают шаги, необходимые для проверки определенной функциональности или сценария использования. Тест-кейсы могут включать в себя ввод данных, выполнение определенных действий пользователем и проверку ожидаемых результатов.

Затем необходимо следовать тест-кейсам, выполняя каждый шаг вручную записывать результаты тестирования. В процессе выполнения тестов могут обнаружиться ошибки, непредвиденное поведение или несоответствия спецификациям. Эти проблемы документируются в баг-репортах с описанием проблемы, шагов для воспроизведения и ожидаемого поведения.

Ручное тестирование позволяет оценить пользовательский интерфейс, взаимодействие с приложением и общее качество пользовательского опыта. Это также дает возможность проверить, соответствует ли приложение ожиданиям пользователей и требованиям заказчика.

Однако ручное тестирование имеет свои ограничения. Оно зачастую является трудоемким и подвержено человеческим ошибкам. Кроме того, повторное выполнение тестов при каждом изменении кода может быть затруднительным и неэффективным, поэтому производится обычно на финальном этапе разработки.

Тест-кейс №1. Переход на главную страницу сайта.

Ожидаемый результат: на экране пользователя будет выведен список всех существующих тарифов

Действия тестировщика:

- 1) Пользователь заходит на главную страницу сайта

Тест-кейс №2. Ввод одного значения в фильтрах тарифов мобильных операторов.

Ожидаемый результат: на экране пользователя будет выведен список всех тарифов, удовлетворяющих условию

Действия тестировщика:

- 1) Пользователь заходит на главную страницу сайта
- 2) Пользователь указывает один из параметров в качестве фильтра для тарифов
- 3) Пользователь нажимает на кнопку “Поиск”

Тест-кейс №3. Ввод нескольких значений в фильтрах тарифов мобильных операторов.

Ожидаемый результат: на экране пользователя будет выведен список всех тарифов, удовлетворяющих условию

Действия тестировщика:

- 1) Пользователь заходит на главную страницу сайта
- 2) Пользователь указывает несколько из параметров в качестве фильтра для тарифов
- 3) Пользователь нажимает на кнопку “Поиск”

Тест-кейс №4. Ввод всех значений в фильтрах тарифов мобильных операторов.

Ожидаемый результат: на экране пользователя будет выведен список всех тарифов, удовлетворяющих условию

Действия тестировщика:

- 1) Пользователь заходит на главную страницу сайта
- 2) Пользователь заполняет все поля фильтров для тарифов
- 3) Пользователь нажимает на кнопку “Поиск”

Тест-кейс №5. Обновление страницы после ввода значений в фильтрах тарифов мобильных операторов.

Ожидаемый результат: на экране пользователя будет выведен список всех тарифов, удовлетворяющих условию

Действия тестировщика:

- 1) Пользователь заходит на главную страницу сайта
- 2) Пользователь заполняет одни или несколько полей фильтров для тарифов
- 3) Пользователь нажимает на кнопку “Поиск”
- 4) Пользователь обновляет страницу

Тест-кейс №6. Переход по ссылке на страницу официального провайдера тарифа.

Ожидаемый результат: пользователь будет перенаправлен на страницу официального провайдера тарифа.

Действия тестировщика:

- 1) Пользователь заходит на главную страницу сайта
- 2) Пользователь заполняет одни или несколько полей фильтров для тарифов

4.2 Юзабилити тестирование

Юзабилити тестирование является важным аспектом тестирования программного обеспечения, направленным на оценку удобства использования и общей удовлетворенности пользователей интерфейсом. Этот процесс помогает выявить проблемы в дизайне интерфейса и функциональности, которые могут затруднять работу пользователей с приложением.

Процесс юзабилити тестирования начинается с определения целей и задач, которые должны быть достигнуты. Это может включать проверку того, насколько легко пользователи могут выполнять ключевые задачи в приложении, оценку общего пользовательского опыта или выявление конкретных проблем с интерфейсом.

Следующим шагом является выбор тестировщиков. Для юзабилити тестирования привлекаются реальные пользователи, представляющие целевую аудиторию приложения. Эти пользователи должны иметь различный уровень опыта и навыков для получения разнообразных отзывов.

Затем разрабатываются сценарии или задачи, которые пользователи должны выполнить. Эти сценарии должны отражать типичные действия,

которые пользователи выполняют в приложении. Например, для веб-сайта интернет-магазина это может быть процесс поиска и покупки товара.

В процессе тестирования пользователи выполняют задачи согласно сценариям, а наблюдатели фиксируют их действия, замечания и реакции. Важно не вмешиваться в процесс выполнения задач, чтобы получить максимально естественные и честные результаты.

После завершения тестирования проводится анализ собранных данных. Наблюдатели оценивают, какие задачи вызвали у пользователей затруднения, какие элементы интерфейса были непонятны или неудобны, и какие аспекты вызвали положительные отзывы.

Результаты анализа представляются в виде отчета, который содержит выявленные проблемы, их описание, а также рекомендации по их устранению. Отчет может включать в себя также положительные аспекты, которые необходимо сохранить и развивать.

На основании отчета вносятся правки в разрабатываемый продукт: дизайн и функциональность приложения, после чего может быть проведен повторный цикл юзабилити тестирования для проверки внесенных улучшений.

Юзабилити тестирование помогает сделать приложение более интуитивно понятным и удобным для конечных пользователей, что в конечном итоге повышает их удовлетворенность и лояльность.

Для юзабилити тестирования веб-сервиса были привлечены 21 человек. После прохождения тестирования возможностей сайта, пользователем было предложено пройти анкетирование, состоящее из 4 категорий и 15 вопросов по этим категориям вопросов:

- 1) Общие вопросы
- Какую общую оценку вы бы дали нашему веб-сайту по шкале от 1 до 5?
- Насколько легко было понять, как использовать наш веб-сайт? (Очень легко, Легко, Нейтрально, Трудно, Очень трудно)

- Насколько вам удалось достичь ваших целей на сайте? (Полностью достиг, В основном достиг, Нейтрально, Частично достиг, Не достиг)
- Сколько времени вам потребовалось, чтобы найти нужную информацию о тарифах? (До 10 секунд, от 10 до 30 секунд, не нашел)

2) Вопросы по интерфейсу

- Как бы вы оценили дизайн и внешний вид сайта? (Очень нравится, Нравится, Нейтрально, Не нравится, Очень не нравится)
- Были ли элементы интерфейса понятны и легко находимы?
- Какую оценку вы бы дали удобству навигации по сайту? (Очень удобно, Удобно, Нейтрально, Неудобно, Очень неудобно)
- Был ли у вас опыт путаницы или затруднений при использовании сайта?

3) Вопросы по функциональности

- Насколько полезными вы нашли предоставленные фильтры для выбора тарифов? (Очень полезно, Полезно, Нейтрально, Мало полезно, Совсем не полезно)
- Насколько подробной и актуальной была информация о тарифах по шкале от 1 до 5?

4) Вопросы по производительности

- Насколько быстро загружались страницы сайта? (Очень быстро, Быстро, Нейтрально, Медленно, Очень медленно)
- Были ли у вас проблемы с производительностью сайта, например, страницы долго загружались или были ошибки?

На рисунках 4–15 представлены результаты анкетирования.

Какую общую оценку вы бы дали нашему веб-сайту по шкале от 1 до 5?

21 ответ

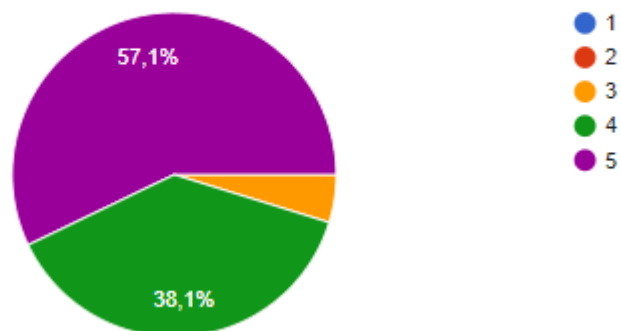


Рисунок 4 — Результаты анкетирования пользователей юзабилити тестирования.

Насколько легко было понять, как использовать наш веб-сайт?

21 ответ

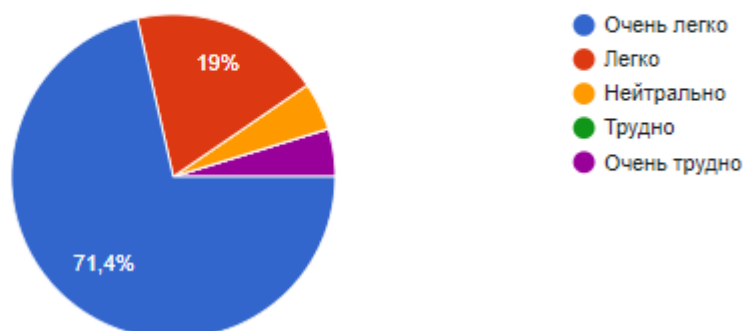


Рисунок 5 — Результаты анкетирования пользователей юзабилити тестирования.

Насколько вам удалось достичь ваших целей на сайте?

21 ответ

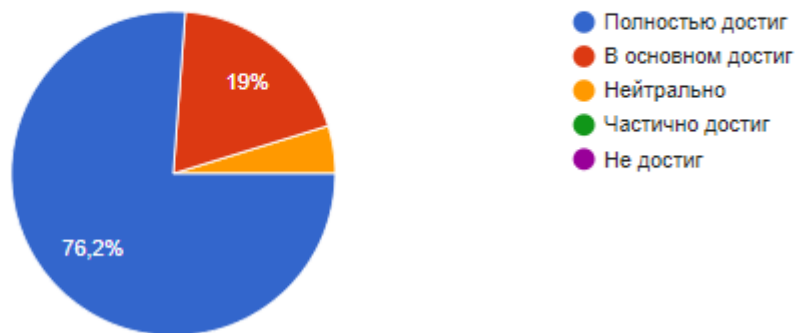


Рисунок 6 — Результаты анкетирования пользователей юзабилити тестирования.

Сколько времени вам потребовалось, чтобы найти нужную информацию о тарифах?

21 ответ

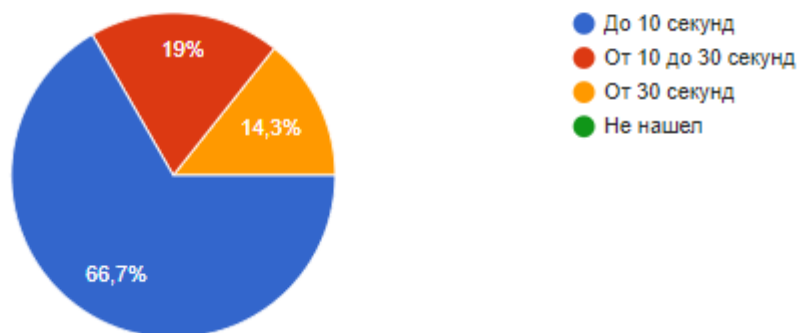


Рисунок 7 — Результаты анкетирования пользователей юзабилити тестирования.

Как бы вы оценили дизайн и внешний вид сайта?

21 ответ

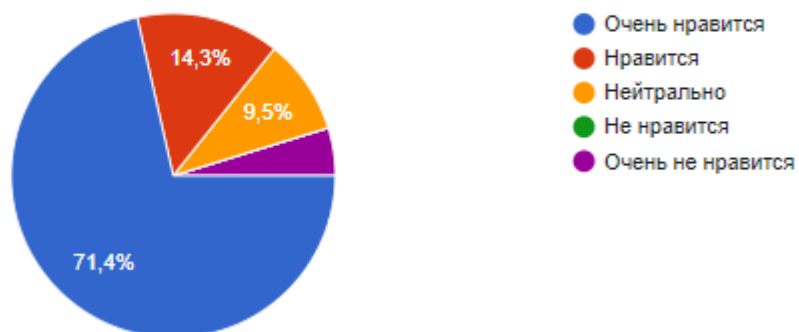


Рисунок 8 — Результаты анкетирования пользователей юзабилити тестирования.

Были ли элементы интерфейса понятны и легко находимы?

21 ответ

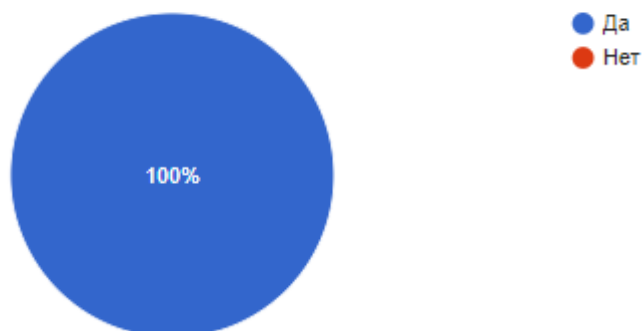


Рисунок 9 — Результаты анкетирования пользователей юзабилити тестирования.

Какую оценку вы бы дали удобству навигации по сайту?

21 ответ

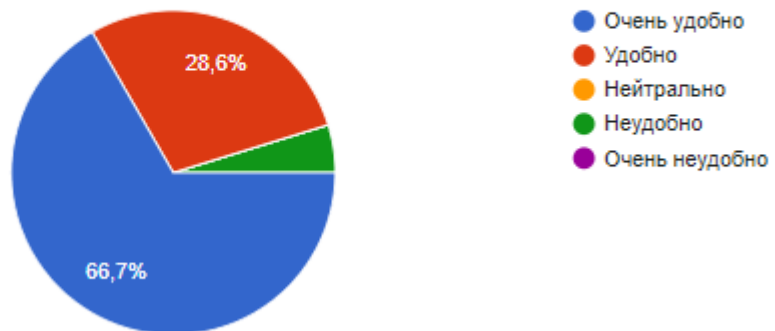


Рисунок 10 — Результаты анкетирования пользователей юзабилити тестирования.

Был ли у вас опыт путаницы или затруднений при использовании сайта?

21 ответ

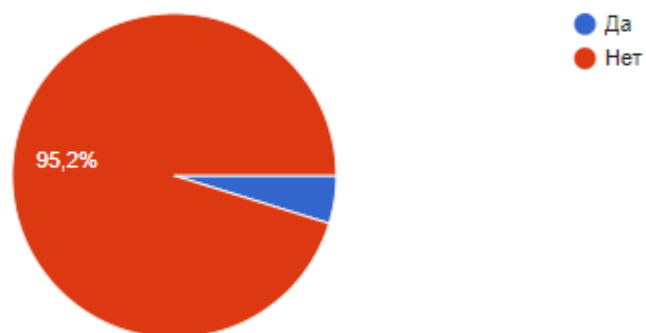


Рисунок 11 — Результаты анкетирования пользователей юзабилити тестирования.

Насколько полезными вы нашли предоставленные фильтры для выбора тарифов?

21 ответ

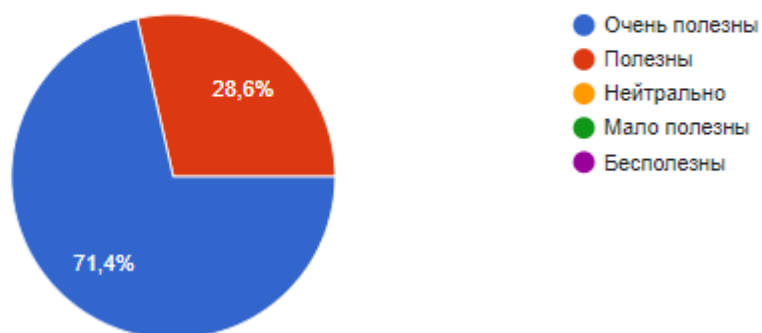


Рисунок 12 — Результаты анкетирования пользователей юзабилити тестирования.

Насколько подробной и актуальной была информация о тарифах?

21 ответ

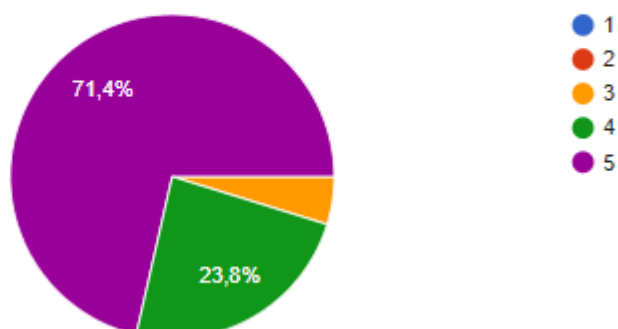


Рисунок 13 — Результаты анкетирования пользователей юзабилити тестирования.

Насколько быстро загружались страницы сайта?

21 ответ

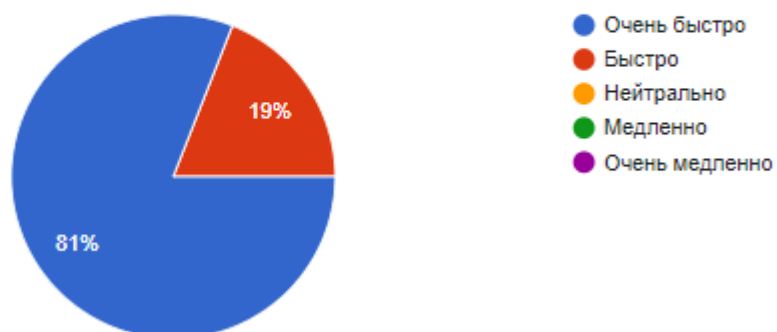


Рисунок 14 — Результаты анкетирования пользователей юзабилити тестирования.

Были ли у вас проблемы с производительностью сайта, например, страницы долго загружались или были ошибки?

21 ответ

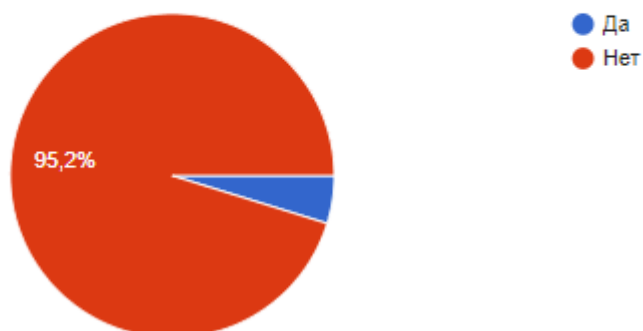


Рисунок 15 — Результаты анкетирования пользователей юзабилити тестирования.

Проведенные юзабилити тестирования веб-сервиса для индивидуального подбора и сравнения тарифов мобильных операторов показали ряд положительных результатов, свидетельствующих о высокой степени удобства и интуитивности использования сайта.

Первым важным аспектом, отмеченным пользователями, является простота и понятность интерфейса. Пользователи высоко оценили логическую структуру и доступность всех основных функций. Интуитивно понятное расположение элементов управления, четкая навигация и единообразный дизайн способствовали легкости освоения сайта даже для тех пользователей, которые впервые им пользовались. Все основные действия, такие как поиск, сравнение и просмотр тарифов, выполнялись быстро и без затруднений.

Вторым положительным результатом стало удобство ввода и редактирования критериев для подбора тарифов. Пользователи отметили, что процесс задания предпочтений, таких как цена, объем интернет-трафика, количество минут разговора и SMS, был простым и понятным. Элементы управления, такие как ползунки и выпадающие списки, позволяли легко

настроить параметры поиска, что делало процесс подбора максимально точным и удобным.

Третьим важным аспектом является высокая скорость загрузки страниц и быстрое действие системы в целом. Пользователи отметили, что результаты поиска и сравнения тарифов отображались мгновенно, что значительно повышало комфорт работы с сайтом. Быстрая реакция на пользовательские запросы и отсутствие задержек в работе системы создавали положительное впечатление о производительности сервиса.

Кроме того, пользователи высоко оценили наличие подробной информации о каждом тарифном плане. Отображение всех необходимых данных, таких как название оператора, стоимость тарифа, объем включенного интернет-трафика, количество минут и сообщений, а также наличие ссылки на сайт провайдера, позволяли пользователям принимать обоснованные решения. Полнота и достоверность предоставляемой информации способствовали доверию к сервису и удобству его использования.

В заключение, результаты юзабилити тестирования продемонстрировали высокую степень удовлетворенности пользователей интерфейсом и функциональностью веб-сервиса. Простота использования, удобство ввода данных, высокая скорость работы и адаптивный дизайн способствовали положительному восприятию сайта и подтверждают успешность проведенной работы по его разработке. Эти результаты являются важным показателем качества и удобства созданного продукта, что подтверждает его готовность к широкому внедрению и использованию.

4.3 Нагрузочное тестирование

Нагрузочное тестирование включает несколько этапов, которые помогают оценить производительность и устойчивость приложения под

высокой нагрузкой. Процесс начинается с планирования, где определяются цели тестирования, объем работы и сценарии нагрузки. Важно понять, какие функции приложения будут проверяться, и установить метрики производительности, такие как время отклика, пропускная способность и использование ресурсов.

Затем создается тестовая среда, которая должна быть максимально приближена к рабочей, чтобы результаты тестирования были точными и репрезентативными. Это включает настройку серверов, баз данных и других компонентов системы. После этого разрабатываются сценарии тестирования, описывающие действия пользователей или системные запросы, которые будут симулироваться во время тестирования. Например, сценарии могут включать авторизацию пользователей, выполнение поисковых запросов, оформление заказов и другие типичные действия.

Для проведения нагрузочного тестирования используются специальные инструменты, такие как Apache JMeter, Gatling, LoadRunner, Locust и другие. Эти инструменты позволяют создавать и выполнять тестовые сценарии, а также собирать и анализировать данные о производительности. После выбора подходящего инструмента, тесты запускаются, и приложение подвергается нагрузке, симулирующей большое количество пользователей или запросов.

В процессе тестирования собираются данные о производительности системы, которые затем анализируются. Этот анализ позволяет выявить узкие места, такие как медленные запросы к базе данных или недостаточная мощность серверов, а также определить, насколько хорошо приложение масштабируется и насколько устойчиво оно работает под нагрузкой.

По результатам анализа разрабатываются рекомендации по улучшению производительности и устраняются выявленные проблемы. Эти улучшения могут включать оптимизацию кода, модернизацию инфраструктуры или изменение архитектуры системы. Повторное тестирование проводится для проверки эффективности внесенных изменений и подтверждения того, что приложение теперь может справляться с предполагаемой нагрузкой.

Нагрузочное тестирование приложения проводилось с помощью приложения Apache JMeter, которое позволило имитировать множество одновременных запросов к серверу.

Цели тестирования:

- Определить максимальное количество одновременных пользователей, которое может обслуживать система без значительного ухудшения производительности.
- Оценить время отклика системы под нагрузкой.
- Выявить возможные узкие места в производительности приложения.

Тестирование проводилось с использованием следующих сценариев:

- Тестирование при низкой нагрузке: 50 одновременных пользователей.
- Тестирование при средней нагрузке: 200 одновременных пользователей.
- Тестирование при высокой нагрузке: 500 одновременных пользователей.
- Пиковое тестирование: 1000 одновременных пользователей.

Каждый сценарий включал в себя следующие операции:

- Загрузка главной страницы.
- Поиск тарифных планов по заданным параметрам.

Результаты тестирования:

1) Тестирование при низкой нагрузке (50 пользователей):

- Среднее время отклика: ~32 мс.
- Максимальное время отклика: 44 мс.
- Ошибок: 0%.

2) Тестирование при средней нагрузке (200 пользователей):

- Среднее время отклика: ~63 мс.

- Максимальное время отклика: 88 мс.
- Ошибок: 0%.

Тестирование при высокой нагрузке (500 пользователей):

- Среднее время отклика: ~99 мс.
- Максимальное время отклика: 127 мс.
- Ошибок: 0%.

Пиковое тестирование (1000 пользователей):

- Среднее время отклика: ~147 мс.
- Максимальное время отклика: 199 мс.

Ошибок: <1%.

Анализ результатов

- Низкая нагрузка: Приложение показало отличные результаты, с быстрым временем отклика и отсутствием ошибок. Это указывает на высокую эффективность при малом количестве пользователей.
- Средняя нагрузка: Время отклика увеличилось, но система продолжала работать стабильно. Это указывает на хорошую производительность при среднем уровне нагрузки.
- Высокая нагрузка: Время отклика значительно увеличилось, но система продолжала стабильно функционировать,
- Пиковая нагрузка: Приложение начало испытывать трудности, время отклика значительно увеличилось, и количество ошибок хоть и незначительно, но возросло. Это указывает на необходимость оптимизации для обеспечения стабильной работы при максимальной нагрузке в будущем.

Результаты нагрузочного тестирования показали, что приложение способно эффективно работать при низкой, средней и высокой нагрузке. И

лишь при достижении пиковых значений одновременных запросов появляются проблемы с работой системы.

Выводы

В процессе тестирования веб-сервиса для индивидуального подбора и сравнения тарифов мобильных операторов были достигнуты значительные результаты, свидетельствующие о высоком уровне реализации проекта. Конструирование программы включало детальное проектирование всех компонентов системы, что позволило создать функционально завершенный и структурированный продукт. В рамках тестирования были проведены всесторонние проверки всех модулей системы, что подтвердило их работоспособность и соответствие заявленным функциональным требованиям. Тесты охватывали различные сценарии использования, что позволило выявить и устранить возможные ошибки и улучшить общую производительность системы.

Апробация сервиса, включающая реальное использование пользователями, продемонстрировала высокую эффективность и удобство разработанного продукта. Пользователи отметили интуитивно понятный интерфейс, быстроту работы и точность подбора тарифов. Это подтверждает, что выбранные архитектурные решения и технологии полностью соответствуют потребностям конечных пользователей и современным стандартам разработки веб-приложений.

Нагрузочные испытания, проведенные с использованием Apache JMeter, показали, что система успешно справляется с разным уровнем нагрузки. При низкой, средней и высокой нагрузке. Время отклика оставалось на приемлемом уровне, без возникновения ошибок, что свидетельствует о хорошо спроектированной архитектуре и эффективном коде.

Таким образом, цели, поставленные в начале работы, были успешно достигнуты, а разработанный сервис готов к полноценной эксплуатации и дальнейшему развитию.

Заключение

В данной работе была поставлена цель разработать веб-сервис для индивидуального подбора и сравнения тарифов мобильных операторов с учетом финансовых предпочтений пользователя. В ходе работы была проведена комплексная реализация проекта, включающая анализ существующих решений, разработку архитектуры системы, выбор технологий, реализацию функционала и проведение тестирования.

На этапе анализа был изучен ряд существующих решений, что позволило выявить их сильные и слабые стороны и на основе этого определить требования к разрабатываемому сервису. Важным этапом стало создание формальной модели проблемной области, что позволило четко определить структуру данных и взаимосвязи между ними. Это обеспечило основу для разработки алгоритмов фильтрации и сравнения тарифов, что, в свою очередь, гарантирует точность и актуальность предоставляемой информации.

Разработка включала создание детализированной архитектуры системы, выбор и интеграцию современных технологий, таких как Spring Boot, Hibernate и FreeMarker. Эти технологии были выбраны не случайно: они обеспечивают высокую производительность, надежность и гибкость системы. Реализация функционала, включающая интеграцию с базами данных операторов, позволила создать эффективный механизм для обработки и предоставления данных пользователям.

Проведенные тесты подтвердили эффективность разработанного веб-сервиса. Нагрузочные испытания показали, что система способна

выдерживать значительные нагрузки, обеспечивая стабильную работу и минимальное время отклика. Выявленные проблемы при пиковых нагрузках указывают на необходимость дальнейшей работы по оптимизации, однако в целом проект достиг поставленных целей и задач.

Разработанный веб-сервис значительно упрощает процесс выбора тарифных планов для пользователей, экономит их время и предоставляет актуальную информацию, что повышает удовлетворенность пользователей и способствует усилению конкуренции среди мобильных операторов. Важно отметить, что проект имеет потенциал для дальнейшего развития. Возможные направления включают добавление новых функций, таких как персонализированные рекомендации на основе анализа поведения пользователей, интеграция с дополнительными источниками данных и улучшение интерфейса для повышения удобства использования.

Таким образом, поставленная цель была достигнута, а разработанный веб-сервис демонстрирует высокую практическую значимость и потенциал для дальнейшего развития.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Морозова, Е. С. Технология создания виртуальных интерактивных туров / Е. С. Морозова, В. В. Лавров // Теплотехника и информатика в образовании, науке и производстве : сборник докладов I Всероссийской научно-практической конференции студентов, аспирантов и молодых ученых (ТИМ2012) с Международным участием / УрФУ [и др.] ; под ред. Н. А. Спирина.– Екатеринбург, 2012.– С. 245-247.
2. Что такое виртуальный тур? [Электронный ресурс]. —Режим доступа : <https://3dturov.net> (дата обращения 28.04.2022).

Приложение А

Справка о результатах проверки выпускной квалификационной работы
на наличие заимствований

Приложение Б
Техническое задание

Приложение В
Руководство системного программиста