Содержание

|  |  |  |  |
| --- | --- | --- | --- |
| Введение | | | 4 |
| 1 | | Основная часть |  |
| 1.1 | Сведения о предприятии: наименование, вид деятельности, форма собственности, организационно-правовая форма предприятия, его тип, адрес, миссия предприятия, его задачи |  |
| 1.2 | Организационная структура предприятия (схема, должностные инструкции руководителей и служащих) |  |
| 1.3 | Инструктаж по технике безопасности |  |
| 1.4 | Основные технико-экономические показатели работы предприятия. Состав оборудования на предприятии |  |
| 2 | | Технологическая часть |  |
| 2.1 | Разработка модели архитектуры информационной системы |  |
| 2.2 | Описание бизнес-процессов заданной предметной области |  |
| 2.3 | Построение модели управления качеством процесса изучения модуля |  |
| 2.4 | Изучение средств автоматизированного документирования |  |
| 2.5 | Построение диаграммы Вариантов использования и диаграммы Последовательности |  |
| 2.6 | Построение диаграммы Кооперации и диаграммы Развертывания |  |
| 2.7 | Построение диаграммы Деятельности, диаграммы Состояний и диаграммы Классов |  |
| 2.8 | Построение диаграммы компонентов |  |
| 2.9 | Построение диаграмм потоков данных |  |
| 2.10 | Стоимостная оценка проекта  Технико-экономическое обоснование |  |
| 2.11 | Построение и обоснование модели проекта |  |
| 2.12 | Установка и настройка системы контроля версий |  |
| 2.13 | Проектирование, разработка и тестирование интерфейса пользователя |  |
| 2.14 | Разработка и тестирование графического интерфейса пользователя |  |
| 2.15 | Реализация и тестирование алгоритмов обработки числовых данных |  |
| 2.16 | Реализация и тестирование алгоритмов поиска |  |
| 2.17 | Реализация и тестирование обработки табличных данных |  |
| 2.18 | Отладка приложения |  |
| 3 | | Выполнение индивидуальных практических работ |  |
| 3.1 | Методы контроля качества в информационных системах. Особенности контроля в различных видах систем |  |
| 3.2 | Мониторинг разработки проекта. Сохранение версий проекта |  |
| 3.3 | Анализ и обеспечение обработки исключительных ситуаций |  |
| 3.4 | Разработка руководства пользователя программного средства по индивидуальному заданию |  |
| Заключение | | |  |
| Список используемых источников | | |  |

2 Технологическая часть

2.1 Разработка модели архитектуры информационной системы

Архитектура разработанной информационной системы направлена на автоматизацию процессов заказа программных решений и поддержку клиентов. Основная цель — предоставить предприятиям удобную веб-платформу для оформления заказов и получения консультаций.

Система имеет следующую архитектурную модель:

* клиентский уровень:

Веб-интерфейс, который позволяет пользователям взаимодействовать с системой. Основные функции интерфейса включают личный кабинет клиента, оформление заказов, доступ к чату поддержки.

Графический интерфейс разработан с учетом удобства использования и интуитивной навигации, чтобы пользователи могли легко находить и заказывать нужные тарифы.

* уровень базы данных:

Для хранения данных используется база данных MySQL, состоящая из двух таблиц: users и orders.

Таблица users содержит поля: ID пользователя (user ID), имя, фамилию, email и пароль. Эта информация необходима для управления учетными записями клиентов и обеспечения безопасности их данных. Таблица orders включает: ID пользователя, который сделал заказ (это позволяет связать заказ с конкретным клиентом через таблицу users), название выбранного тарифа и дату оформления заказа.

Такая структура базы данных позволяет эффективно управлять информацией о клиентах и их заказах, а также легко отслеживать, какие услуги и тарифы выбраны пользователями.

Технологический стек.

При реализации архитектуры были использованы следующие технологии:

Клиентская часть: HTML и CSS для стилизации, JavaScript для интерактивности.

База данных: MySQL для хранения и управления данными о пользователях и заказах.

Данная архитектура позволяет легко управлять данными и поддерживать работу с клиентами, обеспечивая простой и интуитивно понятный процесс оформления заказов и получения поддержки.

2.2 Описание бизнес-процессов заданной предметной области

Основные бизнес-процессы, реализованные на платформе, направлены на упрощение заказа программных решений и предоставление поддержки клиентам. Рассмотрим ключевые этапы взаимодействия пользователя с системой:

Регистрация и авторизация:

Пользователь заходит на сайт и может ознакомиться с информацией о компании и предлагаемых услугах.

Если пользователь решает оформить заказ, он проходит процесс регистрации, где указывает свои личные данные (имя, фамилию, email и пароль). После успешной регистрации пользователь получает доступ к своему личному кабинету, авторизовавшись на платформе.

Выбор и заказ услуги:

В личном кабинете пользователь видит таблицу с четырьмя вариантами тарифов, которые различаются по уровню поддержки и предоставляемым программам.

Ознакомившись с описанием тарифов, пользователь выбирает наиболее подходящий вариант и нажимает кнопку «Заказать».

После оформления заказа с клиентом связывается менеджер для уточнения деталей и дальнейшего обеспечения услуг.

Поддержка клиентов:

В личном кабинете пользователю также доступен чат для оперативной связи с технической поддержкой.

Клиент может использовать чат для решения любых вопросов, связанных с заказом, техническими проблемами или уточнениями по тарифам. Это позволяет улучшить взаимодействие с клиентами и повышает качество обслуживания.

Обработка заказов:

После получения заказа система регистрирует его в базе данных, связывая информацию с учетной записью клиента. Менеджеры обрабатывают заявки и связываются с пользователями для обеспечения дальнейшего выполнения услуги, в том числе интеграции с решениями, предоставляемыми компанией.

Эта последовательность шагов обеспечивает удобный и логичный процесс взаимодействия клиента с платформой, что делает процесс заказа и получения поддержки максимально простым и эффективным.

Такой подход позволяет минимизировать количество действий со стороны клиента, предоставляя все необходимые инструменты в одном месте, что способствует более высокому уровню удовлетворенности пользователей.

2.3 Построение модели управления качеством процесса изучения модуля

Управление качеством является важной частью разработки информационной системы, так как оно обеспечивает соответствие системы ожиданиям пользователей и требованиям бизнеса. В рамках данного проекта предполагается внедрение следующих принципов управления качеством:

Определение стандартов качества:

На начальном этапе необходимо определить критерии, по которым будет оцениваться качество системы. Это могут быть показатели, связанные с удобством использования, временем отклика системы, надежностью и уровнем удовлетворенности клиентов.

Мониторинг и оценка:

Рекомендуется внедрить механизмы мониторинга работы системы, чтобы отслеживать производительность и доступность платформы. Это может включать использование аналитических инструментов для сбора данных о посещаемости сайта, времени загрузки страниц и активности пользователей.

Также важно проводить периодический анализ отзывов пользователей и статистики по выполнению заказов, чтобы выявлять возможные проблемы и области для улучшения.

Обратная связь от пользователей:

Сбор обратной связи от клиентов поможет выявить недостатки в работе системы и улучшить качество обслуживания. Это может осуществляться через опросы, отзывы в чате поддержки или специальные формы на сайте.

Постоянное улучшение:

На основании полученных данных необходимо регулярно проводить анализ и обновление функционала системы для повышения качества услуг. Это включает в себя исправление ошибок, добавление новых функций и оптимизацию процессов.

Эта модель управления качеством направлена на обеспечение высоких стандартов работы системы, что в свою очередь способствует повышению удовлетворенности пользователей и успешному выполнению бизнес-задач.

2.4 Изучение средств автоматизированного документирования

Автоматизированное документирование играет ключевую роль в поддержке и развитии информационных систем. Оно обеспечивает структурированное хранение информации о проекте и позволяет легко находить нужные данные. В рамках разработки веб-платформы для автоматизации бизнес-процессов были использованы следующие инструменты и методы для документирования:

* Документация кода:

Для документирования кода используются комментарии и README-файлы, что позволяет другим разработчикам быстро разобраться в структуре и функционале системы. Такие практики облегчают дальнейшую разработку и сопровождение проекта.

* Документация для пользователей:

Важно подготовить пользовательскую документацию, включая инструкции по регистрации, оформлению заказов и использованию чата поддержки. Это может быть представлено в виде текстовых файлов или онлайн-руководств.

* Инструменты для автоматизированного документирования:

Рассматриваются возможности использования инструментов для автоматизированного документирования, таких как Doxygen или Swagger, которые могут генерировать документацию на основе комментариев в коде и описания API. Это упрощает процесс обновления документации при внесении изменений в код.

* Резервное копирование и хранение документации:

Хранение всей документации в облачных сервисах (например, Google Drive, Dropbox, Mega) обеспечивает доступ к ней из любой точки и минимизирует риск потери данных.

Автоматизированное документирование позволяет эффективно управлять информацией о проекте, что критически важно для поддержания его актуальности и повышения качества обслуживания пользователей.

2.5 Построение диаграммы Вариантов использования и диаграммы Последовательности

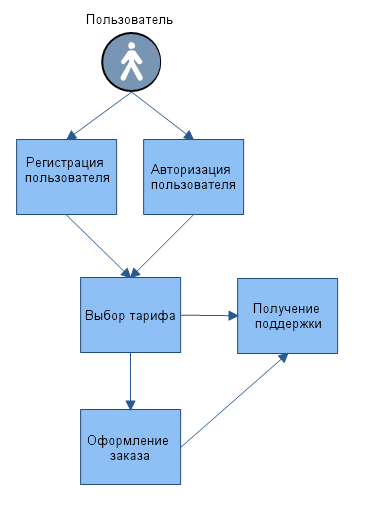


Рис. цифра - Диаграмма Вариантов использования:

Диаграммы Последовательности:

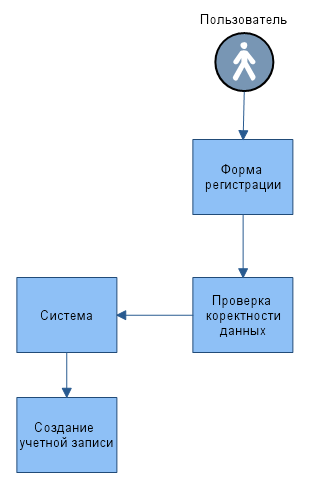


Рис. цифра - Диаграмма последовательности регистрации пользователя

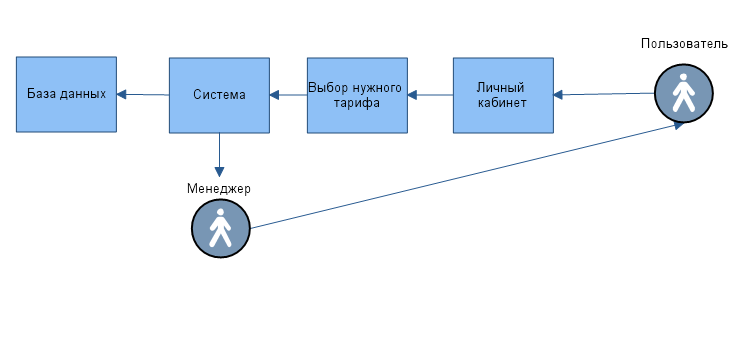


Рис. цифра - Диаграмма последовательности оформления заказа

2.6 Построение диаграммы Кооперации и диаграммы Развертывания

2.10 Стоимостная оценка проекта: Технико-экономическое обоснование

Проект веб-платформы для автоматизации бизнес-процессов и предоставления ИТ-услуг направлен на повышение эффективности работы с клиентами и упрощение процесса заказа. Для технико-экономического обоснования выделим основные статьи затрат и потенциальные выгоды.

* затраты на разработку;

Разработка пользовательского интерфейса: включает создание и настройку веб-страниц с использованием HTML, CSS и JavaScript, а также разработку личного кабинета и функционала чата поддержки.

Настройка базы данных: создание и поддержка таблиц для хранения данных о пользователях и заказах.

Тестирование и отладка: проведение функционального тестирования для обеспечения стабильности работы системы.

Ориентировочные затраты на разработку: В зависимости от объёма работ и используемых инструментов, затраты на начальную разработку составляют примерно 999999999999999999999999 рублей.

* затраты на инфраструктуру;

Хостинг и домен: аренда серверного пространства и покупка домена для обеспечения работы платформы.

Обслуживание базы данных: использование MySQL базы данных и её поддержка.

Резервное копирование и безопасность: дополнительные затраты на обеспечение безопасности данных и регулярное резервное копирование.

Ориентировочные затраты на инфраструктуру: Ежемесячные затраты могут варьироваться в пределах 99999999999999 рублей, в зависимости от выбранного хостинга и требований к безопасности.

* ожидаемые выгоды;

Сокращение времени обработки заказов: автоматизация процессов позволяет сократить трудозатраты сотрудников и ускорить выполнение заказов.

Улучшение качества обслуживания клиентов: с помощью системы чатов и поддержки пользователи могут оперативно получать помощь, что повышает их удовлетворённость.

Рост количества клиентов: удобная платформа и улучшенное взаимодействие с клиентами могут способствовать увеличению базы пользователей и росту доходов компании.

Таким образом, проект способен принести значительные выгоды в плане экономии времени и улучшения качества обслуживания клиентов. Начальные затраты будут окупаться за счёт повышения эффективности и удовлетворенности пользователей.

2.11 Построение и обоснование модели проекта

Цель проекта:

Основная цель проекта — создать веб-платформу для автоматизации процессов заказа ИТ-услуг и поддержки, которая позволит предприятиям легко и быстро оформлять заказы на программные решения и услуги. Проект направлен на повышение эффективности взаимодействия с клиентами и оптимизацию внутренних бизнес-процессов компании.

Выбор архитектуры и технологий

Архитектура проекта: выбрана клиент-серверная модель с клиентской частью, реализованной на HTML, CSS и JavaScript. Данные хранятся на сервере MySQL, что обеспечивает централизованное управление пользовательскими данными и заказами. Архитектура без серверной логики с упрощённой базой данных позволяет быстро развернуть систему и минимизировать затраты на разработку и поддержку.

Выбор технологий:

HTML, CSS, JavaScript обеспечивают простоту и гибкость разработки клиентской части, делая интерфейс доступным для пользователей с различными устройствами.

MySQL используется для хранения данных, так как это проверенное и эффективное решение для работы с реляционными базами данных, способное поддерживать проект на ранних этапах.

Обоснование выбранных решений:

Простота и удобство для пользователей: минималистичный интерфейс с четкими функциями регистрации, выбора тарифа и оформления заказа помогает пользователям интуитивно понять процесс использования платформы.

Экономия ресурсов компании: за счёт автоматизации процесс заказа и поддержки клиентов становятся менее трудоёмкими, что снижает нагрузку на сотрудников.

Потенциал для расширения: в будущем, при необходимости, платформа может быть доработана с добавлением серверной логики и интеграции с 1С для более глубокого анализа данных и учёта заказов.

Модель проекта создана таким образом, чтобы обеспечить минимальные затраты на начальном этапе и гибкость для дальнейшего развития, если платформа начнёт привлекать больше клиентов и заказов.

2.12 Установка и настройка системы контроля версий ?????????

Для управления версиями кода и облегчения совместной работы была выбрана система контроля версий Git. Git — это распределённая система, которая позволяет отслеживать изменения в коде, создавать резервные копии и управлять несколькими версиями проекта. Кроме того, Git предоставляет возможность удобного сотрудничества в команде, что делает его популярным выбором для разработки веб-проектов.

Основные шаги настройки и использования Git в проекте:

Инициализация репозитория:

На начальном этапе работы над проектом был создан локальный репозиторий командой git init, что позволило начать отслеживание изменений в файлах проекта.

Затем репозиторий был подключен к удалённому хранилищу на GitHub, используя команду git remote add origin <URL-репозитория>, чтобы хранить копию проекта в облаке и обеспечить доступность для команды.

Работа с коммитами:

В ходе разработки регулярно выполнялись коммиты для фиксации изменений с использованием команды git commit -m "Описание изменений". Это позволило сохранять версию проекта после каждого значительного обновления и упростило возврат к предыдущим версиям в случае необходимости.

Были приняты правила наименования коммитов, чтобы каждая запись отражала суть внесённых изменений и позволяла быстро ориентироваться в истории изменений.

Использование веток:

Для работы над отдельными функциями или исправлениями создавались новые ветки с помощью команды git branch <название ветки>. Это обеспечило независимость разных частей проекта и позволило интегрировать изменения только после их тщательной проверки.

Ветки объединялись в основную (main) после завершения тестирования с помощью команды git merge, чтобы минимизировать риски и ошибки в финальной версии.

Резервное копирование и управление версиями:

Использование GitHub как удалённого хранилища позволило организовать резервное копирование и хранение всех версий проекта, что защитило данные от случайной потери.

Применение системы контроля версий Git позволило поддерживать порядок в коде, документировать изменения и легко управлять разными версиями проекта. Это особенно важно для проектов, требующих поддержки и возможного масштабирования в будущем.

2.13 Проектирование, разработка и тестирование интерфейса пользователя

Цели проектирования интерфейса

Цель разработки пользовательского интерфейса — создание удобного, интуитивного и минималистичного дизайна, который обеспечит простоту использования платформы для клиентов. В интерфейсе предусмотрены функции регистрации, авторизации, просмотра тарифных планов, оформления заказов и общения с поддержкой.

Ключевые элементы интерфейса:

Главная страница:

Содержит основную информацию о компании и услугах, доступную для всех посетителей сайта, новости и проходящие мероприятия.

Предусмотрены кнопки для регистрации и авторизации, чтобы пользователи могли быстро перейти к личному кабинету.

Кабинет пользователя:

Доступен только для зарегистрированных пользователей после авторизации.

Содержит таблицу с четырьмя тарифными планами, где указаны особенности каждого тарифа, такие как продолжительность поддержки и доступные программные решения.

Кнопка «Заказать» рядом с каждым тарифом позволяет пользователю быстро выбрать нужный вариант.

Чат поддержки:

Предоставляет пользователям возможность связаться с поддержкой, чтобы получить помощь или уточнить детали заказа. Чат доступен в личном кабинете и реализован для оперативного взаимодействия с клиентами.

Процесс разработки интерфейса

HTML использовался для структурирования страниц, обеспечивая удобное расположение элементов интерфейса.

CSS применялся для создания визуального оформления, включая цвета, шрифты и выравнивание элементов, что улучшает восприятие сайта пользователями.

JavaScript обеспечил интерактивные элементы, такие как кнопки для оформления заказа и отправку сообщений в чате.

Тестирование интерфейса

Для проверки удобства использования и функциональности интерфейса были проведены следующие тесты:

Функциональное тестирование: проверка всех кнопок и ссылок на страницах для подтверждения их работы. Например, проверка кнопки «Заказать» для всех тарифных планов, чтобы убедиться, что заказы фиксируются корректно.

Кросс-браузерное тестирование: тестирование интерфейса в различных браузерах (Google Chrome, Mozilla Firefox, Microsoft Edge) для проверки корректного отображения элементов и стабильности работы.

Юзабилити-тестирование: проведение небольшого опроса среди тестировщиков для оценки интуитивности интерфейса и простоты выполнения основных действий.

Разработка интерфейса была выполнена с учетом удобства пользователей, что обеспечивает простоту оформления заказа и удобное взаимодействие с поддержкой.

2.14 Разработка и тестирование графического интерфейса пользователя

Разработка графического интерфейса

Для создания графического интерфейса были выбраны простые и нейтральные цветовые схемы, которые способствуют лёгкости восприятия информации и не отвлекают пользователя от основных функций сайта.

Цветовая палитра:

Основной цвет: тёмный фон для обеспечения хорошей читаемости текста.

Акцентные цвета: использовались желтые и оранжевые цвета для кнопок и выделения элементов интерфейса

Шрифты:

Основной шрифт: простой и легко читаемый, подобранный для того, чтобы обеспечивать удобство восприятия текста на разных устройствах.

Размеры шрифта: основные заголовки имеют более крупный размер для быстрого ориентира, а основной текст — стандартный размер для лёгкого чтения.

Иконки и визуальные элементы:

Использованы небольшие иконки для обозначения функций, таких как чат поддержки и кнопки действий. Иконки помогают пользователям быстро понимать функции, не требуя дополнительных пояснений.

Тестирование графического интерфейса

Для проверки качества графического интерфейса были проведены следующие тесты:

Визуальное тестирование:

Тестирование всех визуальных элементов на соответствие ожидаемому дизайну, включая проверку цветов, шрифтов и выравнивания. Убедились, что интерфейс соответствует выбранной палитре и стилю.

Адаптивное тестирование:

Проверка интерфейса на различных устройствах (настольных компьютерах, планшетах, мобильных устройствах), чтобы убедиться, что все элементы корректно отображаются и интерфейс остаётся удобным при изменении размеров экрана.

Тестирование восприятия:

Проведено среди небольшой группы пользователей, чтобы оценить визуальную привлекательность интерфейса. Пользователи отметили лёгкость восприятия и удобство расположения элементов, что говорит о достижении целей дизайна.

В результате графический интерфейс получился интуитивно понятным, удобным и визуально привлекательным, что способствует лучшему взаимодействию с пользователями и повышению их удовлетворённости.

2.15 Реализация и тестирование алгоритмов обработки числовых данных

Реализация обработки числовых данных

В текущем проекте числовая обработка представлена в ограниченном объёме, так как большинство операций связано с упрощённой базой данных и прямым отображением данных о тарифах. Однако, проект предусматривает базовую обработку числовых данных, включающую:

Подсчёт общей стоимости услуг:

Для каждого тарифного плана отображается фиксированная стоимость услуг, привязанная к уровню поддержки и объёму программного обеспечения. При оформлении заказа выбранная стоимость тарифа сохраняется в базе данных для дальнейшего учёта.

Расчёт времени поддержки:

Тарифы отличаются не только стоимостью, но и продолжительностью поддержки, что учитывается при заказе услуги. В базе данных сохраняется дата заказа и имя тарифа, чтобы служба поддержки могла легко ориентироваться в условиях предоставляемой помощи.

Тестирование алгоритмов обработки числовых данных

Для проверки корректности работы с числовыми данными были выполнены следующие тесты:

Проверка точности расчётов:

Для каждого тарифа проверено, что данные стоимости и времени поддержки корректно сохраняются и отображаются для пользователей. В процессе тестирования выявлены небольшие ошибки отображения, которые были оперативно исправлены.

Тестирование корректности данных при различных условиях:

Тестирование включало моделирование заказов на разные тарифы с последующим просмотром сохранённых данных в базе. Это подтвердило, что данные стоимости и условий поддержки сохраняются без искажений и могут быть использованы для будущих отчётов и анализа.

Нагрузочное тестирование:

Проведено моделирование большого количества заказов для проверки стабильности системы при большом числе записей. Результаты показали, что обработка данных остаётся стабильной, и база данных справляется с загрузкой без существенных задержек.

На данный момент алгоритмы обработки числовых данных реализованы в базовой форме, что соответствует текущим потребностям проекта. При дальнейшем развитии платформы возможна реализация более сложных расчётов, связанных с аналитикой заказов и статистикой использования услуг.

2.16 Реализация и тестирование алгоритмов поиска

Реализация алгоритмов поиска

В текущей версии проекта реализован базовый алгоритм поиска информации в базе данных. Основные операции включают:

Поиск пользователя:

Система позволяет искать пользователя в базе данных по его уникальному идентификатору (user ID), чтобы отобразить данные клиента при оформлении или обработке заказа.

Для повышения удобства информация о пользователе привязывается к заказу через user ID, что позволяет быстро получать нужные данные.

Поиск заказов:

Для каждого пользователя возможен поиск и фильтрация заказов, что помогает отслеживать активные и завершённые заказы, а также анализировать историю взаимодействия с клиентом.

В будущем предусмотрено добавление более продвинутых фильтров (например, поиск по дате, статусу заказа и типу тарифа).

Тестирование алгоритмов поиска

Алгоритмы поиска тестировались для обеспечения корректной и быстрой работы с данными. Основные тесты включали:

Функциональное тестирование:

Проверено, что при вводе user ID система корректно находит соответствующего пользователя и отображает его данные. Тестирование показало стабильные результаты: при корректных данных поиск завершался без ошибок.

Тестирование граничных значений:

Проверен поиск при несуществующем user ID и пустом запросе. В таких случаях система корректно возвращала сообщения об отсутствии данных или предупреждение о необходимости ввода ID.

Оптимизация запросов:

Тестирование скорости выполнения запросов при больших объёмах данных подтвердило, что алгоритмы поиска эффективно обрабатывают даже большое количество записей. Это обеспечено оптимизацией SQL-запросов и индексацией полей user ID и order ID в базе данных.

Алгоритмы поиска обеспечивают удобное и быстрое нахождение данных пользователей и заказов, что упрощает обработку заказов и поддержку клиентов. В дальнейшем функционал поиска может быть расширен для улучшения аналитики и удобства использования.

2.17 Реализация и тестирование обработки табличных данных

Реализация обработки табличных данных

Обработка табличных данных в проекте играет ключевую роль для упрощения взаимодействия пользователей с тарифами и их заказами. Основные элементы обработки данных включают:

Отображение тарифных планов:

В личном кабинете пользователя представлена таблица с четырьмя тарифными планами. Для каждого тарифа указаны его название, уровень поддержки и стоимость, что позволяет пользователю удобно сравнить варианты и выбрать нужный.

Отображение заказов:

Данные о заказах также берутся из базы данных, что позволяет службе поддержки и менеджерам эффективно отслеживать заказы.

Тестирование обработки табличных данных

Для проверки корректного отображения и обработки табличных данных были проведены следующие тесты:

Проверка корректности данных:

При тестировании убедились, что данные о тарифах и заказах отображаются правильно и полностью соответствуют информации в базе данных. Проверка охватывала как корректное заполнение всех полей, так и отображение актуальных данных после обновлений.

Тестирование обновления данных:

Изменения в тарифах и новых заказах проверялись на корректное отображение в реальном времени. Убедились, что изменения в базе данных сразу же отражаются в таблицах на сайте, минимизируя риск некорректного отображения информации.

Проверка пользовательского опыта:

Таблицы были протестированы на удобство использования и восприятие. Внесены некоторые изменения в оформление (размер шрифтов, отступы), чтобы улучшить читаемость данных и сделать работу с таблицами более интуитивной для пользователя.

В результате таблицы с данными о тарифах и заказах обеспечивают удобное и наглядное представление информации для пользователей, что повышает удобство использования сайта и его функциональность.

2.18 Отладка приложения

* процесс отладки;

Отладка приложения является важным этапом разработки, позволяющим выявить и устранить ошибки, а также оптимизировать производительность. В процессе отладки использовались следующие методы и инструменты:

* инструменты разработчика;

В браузерах (Google Chrome и Mozilla Firefox) использовались встроенные инструменты разработчика для отладки JavaScript-кода. Они позволили проверять ошибки в консоли, а также отслеживать выполнение скриптов и взаимодействие с DOM.

В частности, инструменты разработчика использовались для проверки работоспособности функций, связанных с оформлением заказа и взаимодействием с чатом поддержки.

* логирование;

В код добавлены логирующие сообщения, что помогло отслеживать выполнение ключевых функций и выявлять ошибки в процессе выполнения. Логирование использовалось для отображения ошибок и состояний во время работы приложения.

* тестирование сценариев;

Проведено тестирование основных пользовательских сценариев (регистрация, авторизация, оформление заказа) для проверки стабильности и корректности работы всех функций. Это позволило выявить не только ошибки, но и участки, где необходима оптимизация.

Обнаруженные и устранённые ошибки

В процессе отладки были выявлены следующие ошибки и недочёты:

* ошибки в javascript;

При тестировании кнопки «Заказать» были обнаружены ошибки, связанные с некорректным формированием данных для отправки в базу. Эти ошибки были устранены путём добавления дополнительных проверок на наличие необходимых данных перед выполнением операций.

* проблемы с отображением;

Некоторые элементы интерфейса отображались некорректно на мобильных устройствах. После анализа и тестирования было принято решение о внесении изменений в CSS, что улучшило адаптивность интерфейса.

* ошибки в логике работы.

Выявлены ошибки в логике работы чата поддержки, из-за чего сообщения пользователей не всегда доходили до оператора. Проблема была решена путём улучшения обработки событий и добавления дополнительной валидации на стороне клиента.

Рекомендации по отладке

Рекомендуется использовать инструменты разработчика для быстрого поиска ошибок в JavaScript и проверки работоспособности элементов интерфейса.

Добавление логирования может значительно упростить отладку сложных сценариев, позволяя отслеживать, что происходит на каждом этапе выполнения программы.

Регулярное тестирование всех функций приложения на разных устройствах поможет выявить ошибки, связанные с адаптивностью и кросс-браузерностью.

Отладка приложения была проведена успешно, что позволило устранить основные ошибки и повысить стабильность работы платформы.

Заключение

Список используемых источников