Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления
Кафедра интеллектуальных информационных технологий

К защите допустить:

Заведующий кафедрой ИИТ

_____ Д. В. Шункевич

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
на тему:

Информационная справочная система о вине

БГУИР КР 1-40 03 01 02 003 ПЗ

Студент гр. 021732 Д. С. Шаповалов
Руководитель Н. В. Гракова

Минск 2023

СОДЕРЖАНИЕ

Перечень условных обозначений 2
Введение 3
1 Анализ подходов к разработке информационно справочной системы 4
1.1 Анализ информационно справочных систем 4
1.2 Анализ используемых технологий при разработки веб-приложений 7
1.3 Анализ фреймворков используемых в веб-разработке 8
1.4 Сравнение аналогов информационно справочной системы о вине 13
1.5 Вывод 15
2 Проектирование информационно справочной системы по вину . 16
2.1 Вывод 22
3 Разработка информационно справочной системы по вину 24
3.1 Вывод 28
Заключение 30
Список использованных источников 31

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ

Перечень условных обозначений (при необходимости) оформляется в алфавитном порядке

ВВЕДЕНИЕ

Информационные справочные системы – это неотъемлемая часть современного информационного общества, обеспечивающая доступ к знаниям и информации в удобной и структурированной форме. Они играют важную роль в обеспечении пользователей необходимыми данными, помогая им принимать обоснованные решения, выполнять задачи и решать проблемы. Эти системы представляют собой комплекс программных и аппаратных средств, спроектированных для сбора, организации, анализа и предоставления информации, и охватывают разнообразные области, начиная от библиотечных каталогов и заканчивая онлайн-поиском, базами данных и экспертными системами. В науке информационные справочные системы применяются для анализа больших объемов данных и выявления связей между различными явлениями и процессами. Это помогает ученым получать новые знания и раскрывать новые закономерности в разных областях науки. Информационные справочные системы являются важным инструментом для обработки и анализа больших объемов информации в различных областях. Они предоставляют быстрый и точный доступ к актуальным данным, что способствует принятию обоснованных решений и решению сложных задач. В данном курсовой работе рассмотрена сущность информационных справочных систем, их роль в современном мире и основные характеристики, которые определяют их функциональность и эффективность.

В рамках курсовой работы, освещающей предметную область "вино необходимо выполнить следующие задачи:

– Провести анализ особенностей данной предметной области, включая изучение периода, связанного с виноделием. Также рекомендуется рассмотреть различные системы, взаимодействующие с этой предметной областью;

– Проанализировать методы и программно-инструментальные средства, предоставляющие возможность создания веб-интерфейса, реализации нечеткого поиска и работы с базами данных в контексте виноделия;

– Выделить технические требования, необходимые для достижения поставленной цели, и разработать соответствующее техническое решение задачи;

– Реализовать спроектированное решение, используя выбранные инструменты и технологии;

– Провести тестирование функционала системы на различных устройствах и браузерах для обнаружения и устранения возможных ошибок и недочетов;

– Предусмотреть возможность масштабирования системы и добавления новой информации по мере расширения предметной области.

1 АНАЛИЗ ПОДХОДОВ К РАЗРАБОТКЕ ИНФОРМАЦИОННО СПРАВОЧНОЙ СИСТЕМЫ

1.1 Анализ информационно справочных систем

Под **информационно-справочной системой (ИСС)** [1] понимается компьютерное программное средство, предназначенное для хранения и предъявления пользователю разнообразной информации справочного содержания. Для ИСС характерны иерархическая организация материала и быстрый поиск информации по различным параметрам.

В наше время большинство интеллектуальных справочных систем ориентированы на веб-платформы. Это связано с растущей популярностью интернета как основного ресурса для поиска информации и решения задач. Веб-приложения, в свою очередь, являются программами, работающими в браузере пользователя и обращающимися к серверу для получения данных.

Веб-приложение - это программное обеспечение, которое функционирует через веб-браузер пользователя[2]. Оно хранится на удаленном сервере и обычно не требует установки на компьютер пользователя. Пользователь взаимодействует с веб-приложением через интерфейс браузера, отправляя запросы и получая ответы от сервера.

Веб-приложения используются для различных целей:

– Доступ к данным и информации: Они предоставляют доступ к разнообразным данным и ресурсам в интернете. Например, почтовые сервисы, социальные сети, новостные порталы, онлайн-банкинг и многие другие сервисы функционируют как веб-приложения;

– Доступ к данным и информации: Они предоставляют доступ к разнообразным данным и ресурсам в интернете. Например, почтовые сервисы, социальные сети, новостные порталы, онлайн-банкинг и многие другие сервисы функционируют как веб-приложения;

– Онлайн-коммуникация и социальные взаимодействия: С помощью веб-приложений пользователи могут обмениваться сообщениями, изображениями, видео, аудиозаписями и поддерживать социальные связи;

– Управление задачами и проектами: Веб-приложения предлагают инструменты для планирования, организации и отслеживания рабочих задач и проектов.

Хорошее веб-приложение должно предоставлять пользователю удобство и эффективность в использовании. Вот ключевые аспекты, которые определяют качество веб-приложения:

Во-первых, интуитивный интерфейс играет важную роль. Пользователи должны легко понимать, как взаимодействовать с приложением с

первого взгляда. Элементы управления должны быть расположены логично, обеспечивая естественное и интуитивное взаимодействие.

Отзывчивость - еще один ключевой аспект. Приложение должно реагировать на действия пользователя моментально, обеспечивая быструю загрузку страниц и оперативное выполнение команд. Это создает позитивный пользовательский опыт.

Доступность к приложению с различных устройств и браузеров - также важный аспект. Приложение должно корректно отображаться и работать как на компьютерах, так и на мобильных устройствах, а также в разных браузерах.

Безопасность - неотъемлемый элемент. Пользователи должны иметь уверенность в защите своих данных. Это включает в себя меры по предотвращению несанкционированного доступа, шифрованию и другие технические решения.

Высокая производительность приложения - еще одно важное требование. Эффективность работы приложения, минимизация времени ожидания и оперативное реагирование на запросы пользователя являются приоритетами.

Надежность - еще один критический аспект. Пользователи ожидают стабильную и бесперебойную работу приложения. Ошибки и сбои должны быть минимизированы.

Хорошо структурированная база данных необходима для эффективного хранения и доступа к данным. Это особенно важно для приложений, работающих с большими объемами информации.

Кроме того, приложение должно быть доступным для всех категорий пользователей, включая тех, кто использует вспомогательные технологии или имеет особые потребности.

Масштабируемость приложения важна с ростом числа пользователей и объема данных. Приложение должно быть готово к увеличению нагрузки.

Наконец, регулярные обновления и поддержка со стороны разработчиков обеспечивают долгосрочную жизнеспособность приложения и удовлетворение потребностей пользователей. В совокупности, эти аспекты определяют качество веб-приложения и обеспечивают положительный пользовательский опыт.

Отзывчивость - еще один ключевой аспект. Приложение должно реагировать на действия пользователя моментально, обеспечивая быструю загрузку страниц и оперативное выполнение команд. Это создает позитивный пользовательский опыт.

Доступность к приложению с различных устройств и браузеров - также важный аспект. Приложение должно корректно отображаться и работать как на компьютерах, так и на мобильных устройствах, а также в разных браузерах.

Безопасность - неотъемлемый элемент. Пользователи должны иметь уверенность в защите своих данных. Это включает в себя меры по предотвращению несанкционированного доступа, шифрованию и другие технические решения.

Высокая производительность приложения - еще одно важное требование. Эффективность работы приложения, минимизация времени ожидания и оперативное реагирование на запросы пользователя являются приоритетами.

Надежность - еще один критический аспект. Пользователи ожидают стабильную и бесперебойную работу приложения. Ошибки и сбои должны быть минимизированы.

Хорошо структурированная база данных необходима для эффективного хранения и доступа к данным. Это особенно важно для приложений, работающих с большими объемами информации.

Кроме того, приложение должно быть доступным для всех категорий пользователей, включая тех, кто использует вспомогательные технологии или имеет особые потребности.

Масштабируемость приложения важна с ростом числа пользователей и объема данных. Приложение должно быть готово к увеличению нагрузки.

Наконец, регулярные обновления и поддержка со стороны разработчиков обеспечивают долгосрочную жизнеспособность приложения и удовлетворение потребностей пользователей. В совокупности, эти аспекты определяют качество веб-приложения и обеспечивают положительный пользовательский опыт.

Веб-приложения могут быть не только сложными, но и иметь простой характер, как, например, показано на примере медицинского приложения на изображении.

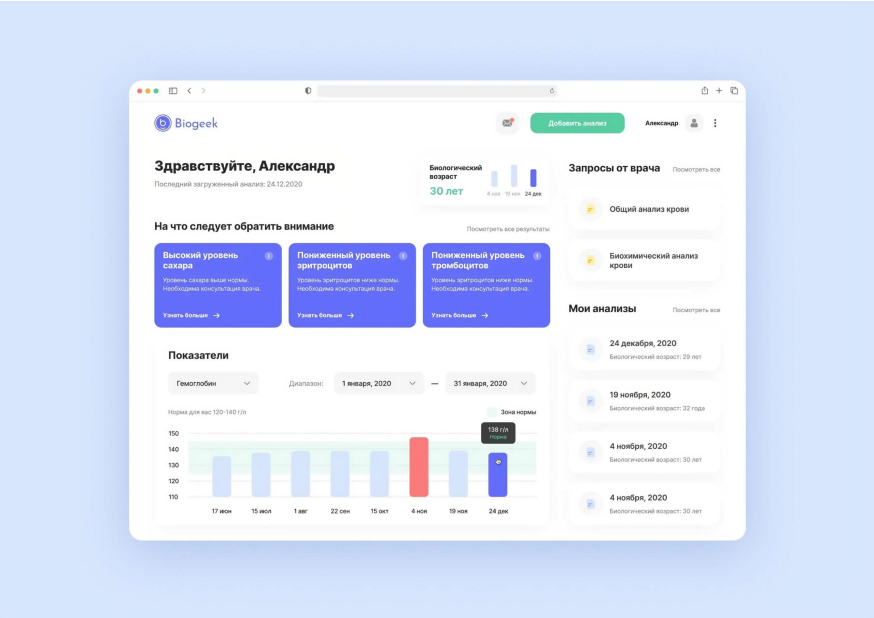


Рисунок 1.1 – Пример веб-приложения

1.2 Анализ используемых технологий при разработки веб-приложений

Технологии веб-разработки включают в себя различные языки программирования и инструменты, применяемые для создания динамичных и полнофункциональных веб-приложений[3]. Веб-приложения представляют собой систему, в которой клиентская и серверная разработки играют ключевую роль. Эти направления разработки включают в себя front-end (технологии, отвечающие за интерфейс) и back-end (технологии, связанные с серверной частью).

Front-end технологии используются для создания интерактивных компонентов веб-сайта, а также элементов, с которыми пользователи взаимодействуют. К ним относятся настройка цветов, стилей текста, вставка изображений, создание кнопок и меню навигации.

Back-end разработка отвечает за создание и управление серверной частью веб-приложения. Эта область включает в себя работу с базами данных, обработку запросов от клиентов, аутентификацию, обновление данных и многое другое.

Одним из основных инструментов back-end разработчика является язык программирования, такой как Python, Java, Ruby, Node.js и другие. Эти языки позволяют создавать логику приложения, обрабатывать данные и взаимодействовать с базами данных.

Кроме того, back-end включает в себя работу с серверами и хостингом, что обеспечивает стабильную работу веб-приложения и его доступность для пользователей в сети Интернет.

Можно рассмотреть иллюстрацию клиент-серверной архитектуры в

графическом виде.(Рис 1.2)

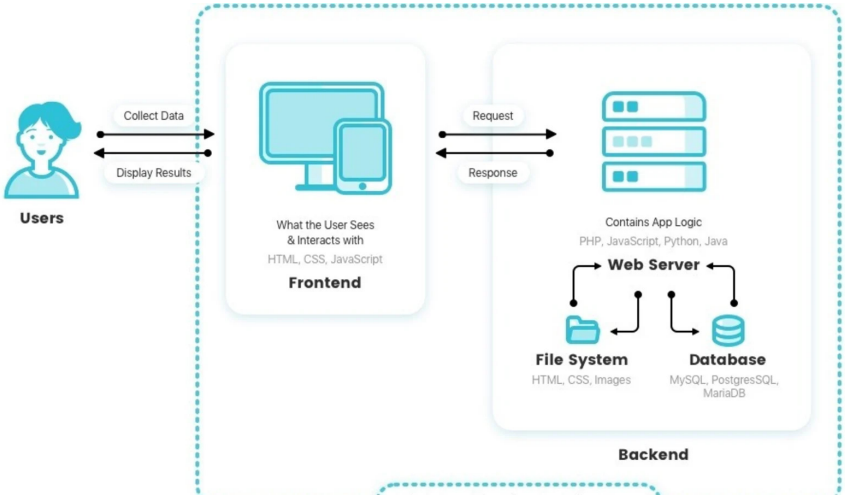


Рисунок 1.2 – Пример клиент-серверной архитектуры

1.3 Анализ фреймворков используемых в веб-разработке

Фреймворки представляют собой инструменты программного обеспечения, упрощающие создание и поддержку сложных или масштабных проектов.

Обычно фреймворк включает только базовые программные модули, а специфичные для проекта компоненты разрабатываются самим разработчиком с использованием этой базы. Это позволяет добиться не только высокой скорости разработки, но и обеспечить большую производительность и надежность решений.

Веб-фреймворк представляет собой платформу для создания сайтов и веб-приложений, упрощающую процесс разработки и интеграцию различных компонентов большого программного проекта в единую систему[4]. Благодаря широким возможностям в реализации бизнес-логики и высокой производительности, такая платформа особенно эффективно применяется для создания сложных сайтов, бизнес-приложений и веб-сервисов.

Использование фреймворков при разработке программных продуктов с точки зрения бизнеса считается наиболее эффективным и качественным подходом. Разработка без применения платформы может быть оправдана только в двух случаях: если проект является очень простым и не предполагает дальнейшего расширения, или если он очень нагружен и требует низкоуровневой оптимизации, например, веб-сервисы с десятками или сотнями тысяч запросов в секунду.

В остальных случаях применение программных платформ, особенно фреймворков, является более быстрым и качественным решением. Фреймворки особенно полезны для проектов с сложной бизнес-логикой и высокими требованиями к скорости работы, надежности и безопасности.

Кроме того, существуют фреймворки как для фронтенд-части (пользовательского интерфейса) так и для бэкенд-части (серверной логики) приложений.

Фреймворки для фронтенда представляют собой набор готовых инструментов, библиотек и шаблонов, которые облегчают и ускоряют процесс разработки пользовательского интерфейса веб-приложений. Они предоставляют разработчикам структуру и организацию для построения компонентов и управления состоянием приложения. Вот некоторые ключевые функции и возможности, которые предоставляют фреймворки для frontend:

– Компонентная архитектура: Фреймворки разбивают пользовательский интерфейс на небольшие компоненты, которые можно разрабатывать, тестировать и поддерживать независимо друг от друга. Это способствует повторному использованию кода и облегчает масштабирование проектов;

– Управление состоянием: Фреймворки предоставляют средства для эффективной организации и управления состоянием приложения. Это позволяет отслеживать изменения данных и их визуализацию в интерфейсе;

– Маршрутизация: Фреймворки предоставляют инструменты для управления навигацией в приложении, что позволяет создавать одностраничные приложения (SPA) с плавной сменой контента без перезагрузки страницы;

– Взаимодействие с сервером: Фреймворки предоставляют средства для отправки и обработки запросов к серверу, что позволяет приложению взаимодействовать с бэкенд-частью;

– Модульность и расширяемость: Фреймворки часто предоставляют механизмы для подключения дополнительных модулей и плагинов, что позволяет расширять функциональность приложения.

Фреймворки для frontend значительно упрощают и ускоряют процесс разработки веб-приложений, позволяя разработчикам сосредотачиваться на создании интерфейса и пользовательского опыта, а не на рутинных задачах организации кода.

Фреймворки для бэкенда предназначены для разработки серверной части веб-приложений. Они предоставляют набор инструментов, библиотек и структур для обработки запросов от клиентской части, управления базами данных, реализации бизнес-логики и обеспечения безопасности приложения [5]. Вот основные функции и возможности, которые предоставляют фреймворки для бэкенда:

– Маршрутизация и обработка запросов: Фреймворки предоставляют средства для определения маршрутов, обработки HTTP-запросов и взаимодействия с клиентской частью приложения;

– Работа с базами данных: Они облегчают создание, подключение и управление базами данных, а также предоставляют ORM (Object-Relational Mapping) для удобной работы с данными;

– Аутентификация и авторизация: Фреймворки предоставляют инструменты для реализации систем аутентификации и авторизации пользователей, обеспечивая безопасность приложения;

– Обработка ошибок и исключений: Они предоставляют механизмы для обработки ошибок и исключений, что повышает надежность приложения;

– Работа с API и внешними сервисами: Фреймворки упрощают интеграцию с внешними API и сервисами, такими как социальные сети, платежные системы и другие;

– Безопасность: Предоставляют средства для защиты от распространенных атак, таких как атаки XSS, CSRF и SQL-инъекции.

Уровень популярности фреймворков можно оценить, обратив внимание на графическое представление данных, представленное на (Рис 1.3)

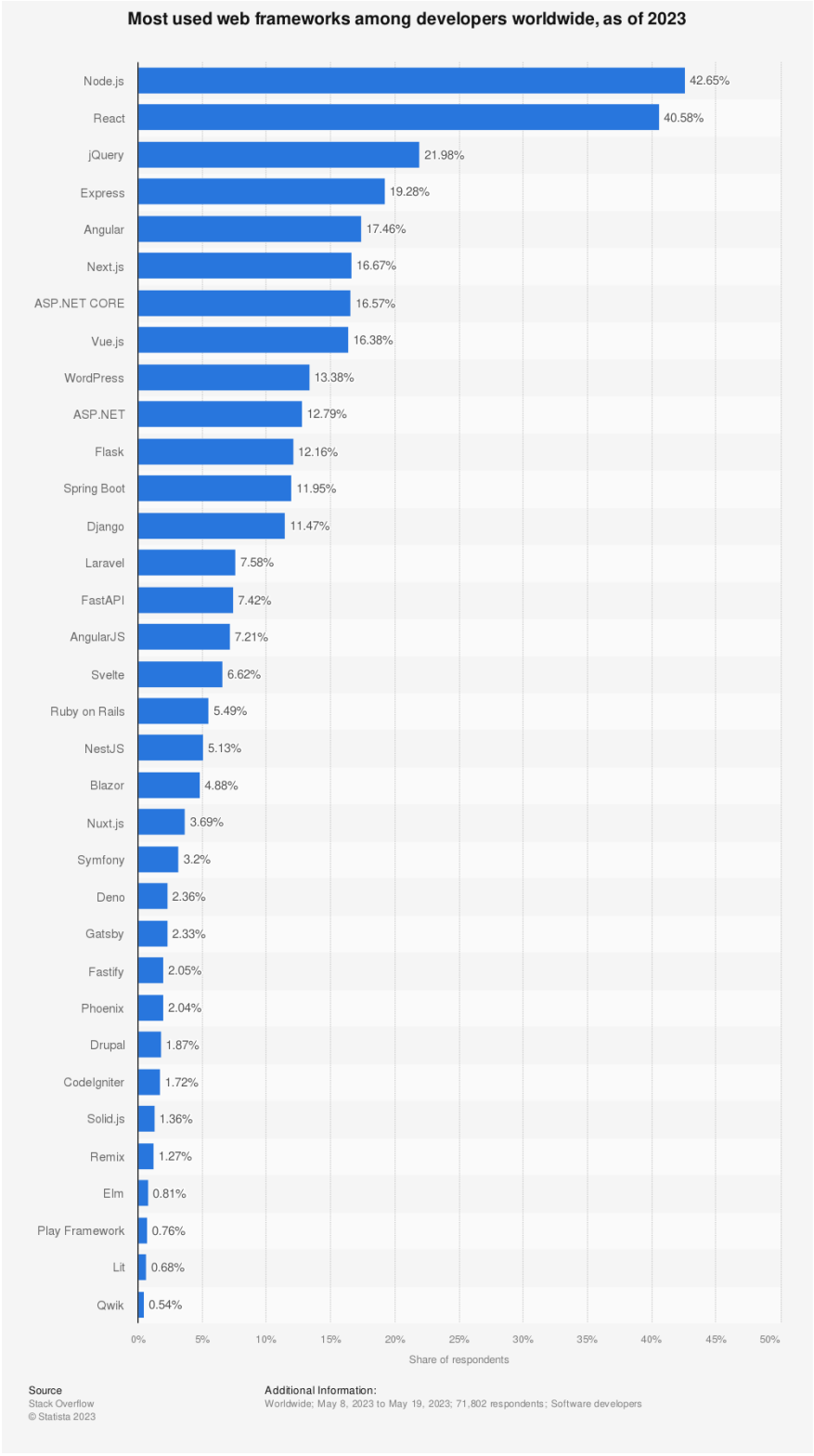


Рисунок 1.3 – Популярность front-end фреймворков

Фреймворки для веб-разработки представляют собой наборы предопределенных инструментов, библиотек и правил, которые упрощают и ускоряют процесс создания веб-приложений. Их использование приносит несколько важных преимуществ:

– Ускорение разработки: Фреймворки предоставляют готовые ком-

поненты и шаблоны, что позволяет программистам сосредотачиваться на специфических задачах проекта, минуя рутинные и стандартные этапы разработки;

– Стандартизация кода: Фреймворки предлагают определенные паттерны и подходы, способствуя созданию чистого, структурированного и согласованного кода. Это улучшает читаемость и понимание кода для команды разработчиков;

– Безопасность: Многие фреймворки предоставляют встроенные механизмы безопасности, такие как защита от атак типа SQL-инъекций, кросс-сайтовых сценариев (XSS) и других потенциальных угроз;

– Масштабируемость: Фреймворки предлагают архитектурные принципы, которые облегчают масштабирование приложений. Это важно для проектов, требующих дальнейшего роста и развития;

– Сообщество и поддержка: Популярные фреймворки имеют большие сообщества разработчиков. Это означает, что вы можете находить решения для множества проблем, а также обмениваться опытом с другими участниками;

– Тестирование: Многие фреймворки предоставляют средства для автоматизированного тестирования, что позволяет обнаруживать и устранять ошибки на ранних этапах разработки;

– Обновления и поддержка: Фреймворки обычно активно поддерживаются и обновляются. Это обеспечивает ваш проект современными технологиями и защитой от уязвимостей;

– Улучшение производительности: Использование фреймворков может увеличить производительность разработки и качество конечного продукта благодаря оптимизации кода и использованию эффективных архитектурных решений;

– Работа в команде: Фреймворки способствуют стандартизации кода, что упрощает совместную работу нескольких разработчиков над одним проектом;

– Экосистема инструментов: Фреймворки обычно поставляются с различными дополнительными инструментами, такими как пакетные менеджеры, средства сборки, библиотеки и плагины, что облегчает разработку и расширение функциональности приложения.

В целом, использование фреймворков в веб-разработке помогает создавать более надежные, эффективные и масштабируемые веб-приложения, сокращая затраты времени и ресурсов на разработку.

1.4 Сравнение аналогов информационно справочной системы о вине

В ходе анализа информационно-справочной системы о вине было замечено, что большинство аналогичных систем имеют простой и минималистичный интерфейс. Основные элементы таких систем включают в себя строку поиска, предназначенную для удобного поиска информации о различных видах вина, а также категории, которые помогают организовать и классифицировать эту информацию.

Возьмем в рассмотрение сайт, изображенный на рисунке 1.4. Он отличается светлым дизайном, не перегруженным сложными графическими элементами и стилями. Информация на сайте представлена в ясном и понятном стиле, с акцентом на удобство пользователя. Единственным элементом, который может привлекать внимание, являются заголовки, оформленные особыми шрифтами. Этот выбор делает их более выразительными и облегчает навигацию по сайту.

Исходя из такого простого и функционального дизайна, можно сделать вывод, что главной целью данного сайта является предоставление пользователю быстрого и удобного доступа к информации о вине. Он не замыкается на создании сложных визуальных эффектов, а вместо этого сосредотачивается на предоставлении информации в доступной форме.

Ключевая аудитория подобных сайтов, вероятно, включает в себя людей, увлекающихся историей вина, а также профессиональных сомелье, которые ищут информацию о различных сортах и характеристиках вин. Упрощенный и интуитивно понятный интерфейс помогает этой аудитории быстро и эффективно находить необходимую информацию о вине, облегчая им работу и удовольствие от исследования этой темы.

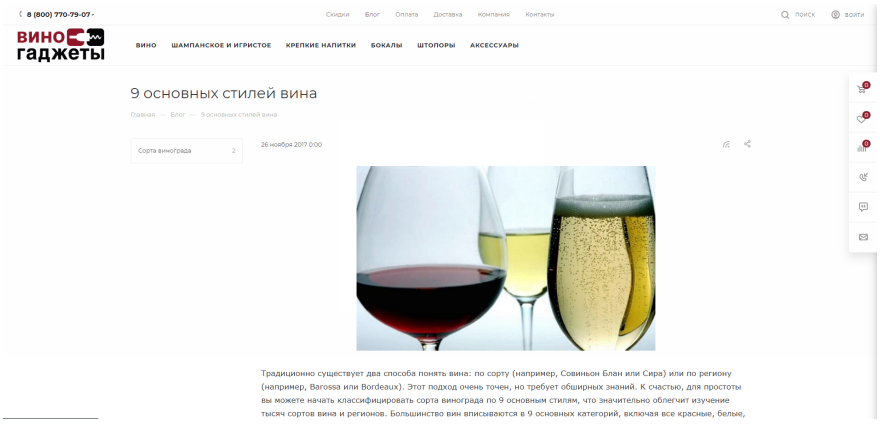


Рисунок 1.4 – Дизайн сайта вино-гаджеты

Стоит также отметить ещё один веб-сайт, а именно - "Винный класс". Этот сайт отличается от предыдущего не только своим богатым дизайном, но и использованием цветовой палитры, которая акцентирует внимание на

теме вина. Эта игра красок напрямую ассоциируется с миром виноделия и может значительно увеличить интерес пользователей к сайту.

На "Винном классе"можно выделить несколько заметных дизайнерских решений. Во-первых, навигационная панель выполнена в ярко-малиновых тонах, что создает ассоциацию с красными винами. Этот цвет привлекает внимание и сразу сообщает посетителям тематику сайта.

Задний фон сайта также выполнен в темно-малиновом цвете, что подчеркивает атмосферу таинственности и элегантности, характерной для многих сортов вин. Этот выбор цвета создает уютное и привлекательное визуальное окружение.

Основной блок, предназначенный для представления информации о винах, выполнен в бежевых тонах. Это позволяет информации о винах выделяться и быть более читаемой на фоне темных оттенков сайта. Бежевый цвет также ассоциируется с цветом винной кожи и дополняет общий стиль сайта.

Кроме того, на "Винном классе"пользователи могут найти не только информацию о конкретных видах вина, но и данные о содержании спирта, уровне сахара, углекислоте и других характеристиках вин. Это дополнительные возможности, которые делают сайт более информативным и полезным для тех, кто интересуется виноделием.

"Винный класс"который представлен на 1.5 демонстрирует собой прекрасный пример того, как дизайн сайта может подчеркнуть его тематику и привлечь внимание аудитории, создавая атмосферу, соответствующую миру вина. Этот сайт может привлечь как профессиональных сомелье, так и любителей вина, предоставляя им информацию и удовольствие от исследования этой увлекательной темы.

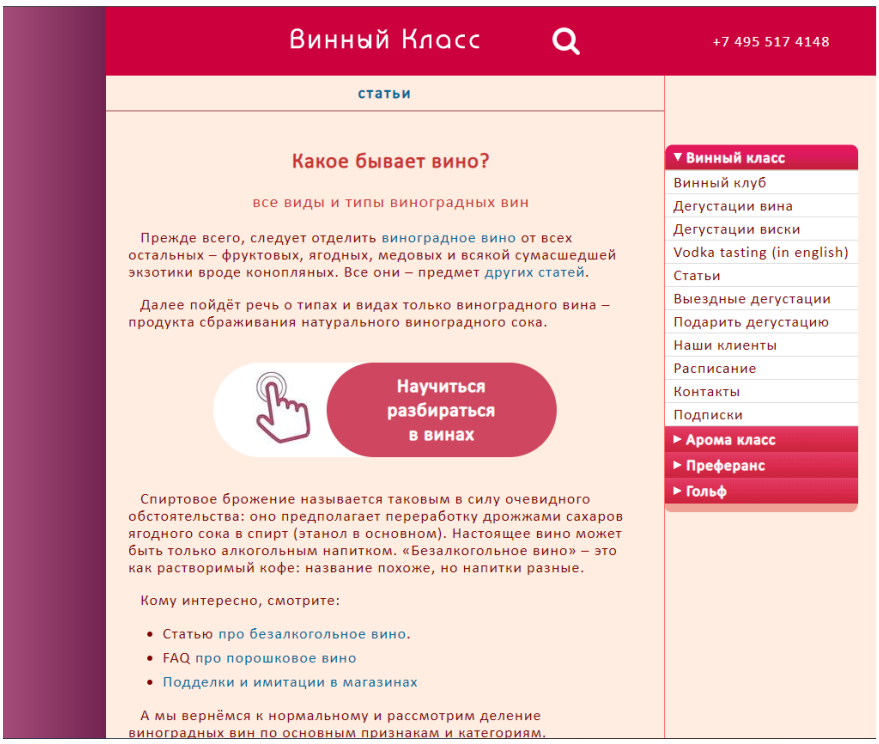


Рисунок 1.5 – Дизайн сайта Винный класс

1.5 Вывод

После проведенного анализа можно сделать следующие выводы:

– Информационные справочные системы, как правило, представляют собой онлайн-ресурсы, которые могут быть доступны через веб-сервисы, веб-сайты или веб-приложения.

– При создании веб-приложений используется клиент-серверная архитектура, где применяются технологии как на стороне клиента (front-end), так и на стороне сервера (back-end).

– Для эффективной разработки широко применяются фреймворки, которые представляют собой готовый инструментарий для упрощения процесса разработки приложений.

– В веб-разработке базы данных играют важную роль, и выбор конкретной базы данных зависит от требований и характера проекта.

– Кроме того, следует отметить, что безопасность данных и пользовательская дружелюбность интерфейса также имеют ключевое значение при разработке информационных справочных систем.

2 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННО СПРАВОЧНОЙ СИСТЕМЫ ПО ВИНУ

Предметная область по предоставлению информации о вине включает в себя всё, что связано с алкогольным напитком. Это включает в себя информацию о различных видах вина, их истории, производстве, регионах, сортах винограда, а также оценках и рейтингах вин. Системы в предметной области виномании (информации о вине) могут быть очень полезными и удовлетворить разнообразные потребности пользователей. Одна из таких потребностей это образование. Данная система будет предоставлять образовательный контент о вине, включая историю, производственные процессы, сорта винограда и регионы производства. Это полезно для начинающих виноманов и всех, кто хочет углубить свои знания. Вторая потребность это рейтинг, пользователи могут получать рекомендации по выбору вина на основе своих предпочтений и вкусов. Система будет предоставлять рейтинги и отзывы от экспертов и других пользователей, помогая принимать информированные решения. Данная система ориентирована на аудиторию, которая будет представлять собой, как новичков которые начинают интересоваться вином и ищущие информацию о его основах, так и коллекционеры которые коллекционируют вино и ищут способы эффективного управления своей коллекцией.

Большинство современных информационных систем используют клиент-серверную архитектуру. Эта архитектура представляет собой организацию системы, в которой процессы запроса и предоставления услуг разделены между несколькими компьютерами, соединенными в сеть. В контексте клиент-серверной архитектуры, компьютеры разделяются на два основных типа: клиентские и серверные.

Один из ключевых аспектов клиент-серверной архитектуры - это разделение обязанностей между клиентом и сервером. Клиентская машина часто имеет дружественный и интуитивно понятный интерфейс для удобства пользователей. Сервер, с другой стороны, занимается обработкой запросов, хранением данных и выполнением вычислений, необходимых для обеспечения работы системы.

В данном приложении также используется клиент-серверная архитектура, что позволяет эффективно управлять данными и предоставлять услуги пользователям. Пример работы этой архитектуры можно увидеть на практике, где клиентская часть обеспечивает простое и понятное взаимодействие с сервером, обеспечивая эффективную работу всей системы. Пример данной архитектуры можно увидеть на (Рис. 2.1)

Клиентские компьютеры являются инициаторами запросов. Они отправляют запросы на централизованную служебную машину, которая на-

зывается сервером. Сервер, в свою очередь, обрабатывает эти запросы и предоставляет нужные услуги или данные. Этот процесс взаимодействия между клиентом и сервером позволяет клиентам получать доступ к ресурсам и функциям, предоставляемым сервером, с минимальными затратами на обработку данных на стороне клиента.

Преимущества клиент-серверной архитектуры включают масштабируемость, централизацию данных и ресурсов, а также легкость обновления и управления системой. Она широко применяется в различных областях, включая веб-приложения, базы данных, облачные сервисы и многое другое.

В данном приложении также используется клиент-серверная архитектура, что позволяет эффективно управлять данными и предоставлять услуги пользователям. Пример работы этой архитектуры можно увидеть на практике, где клиентская часть обеспечивает простое и понятное взаимодействие с сервером, обеспечивая эффективную работу всей системы.

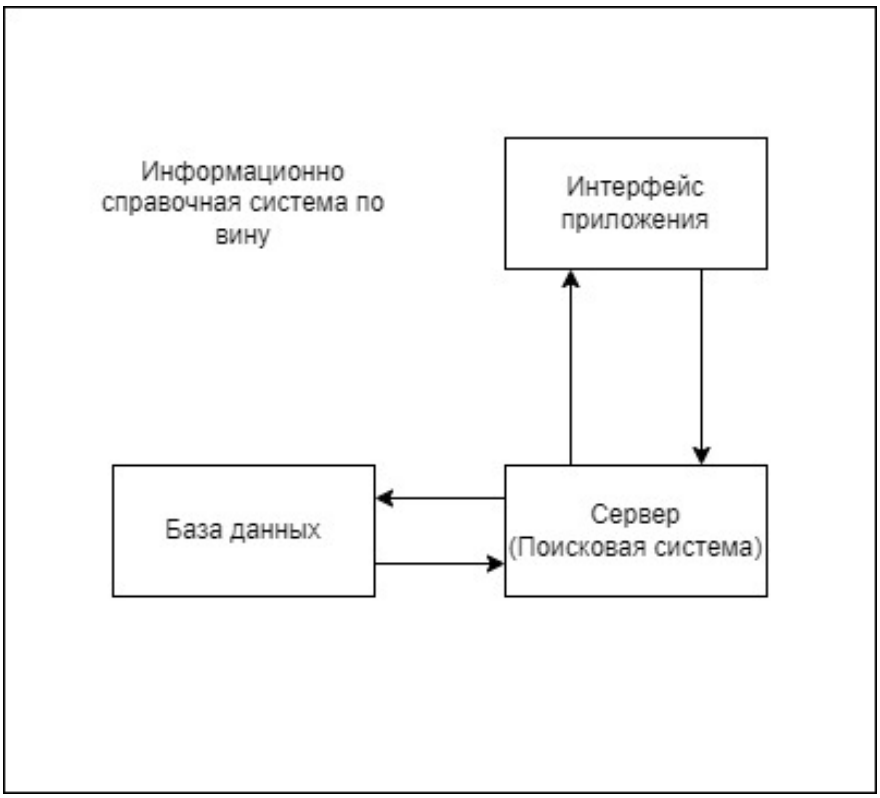


Рисунок 2.1 – Архитектура приложения

Клиент-серверное приложение представляет собой мощную систему, предназначенную для удовлетворения потребностей пользователей в поиске и получении информации о винах. Эта система обеспечивает пользователям простой и эффективный способ взаимодействия с базой данных вин, и позволяет получать подробную информацию о различных видах вина.

Основной функционал клиент-серверного приложения включает в себя:

– Поисковую строку: Пользователь может ввести в поисковую строку различные запросы, связанные с вином. Например, он может ввести название конкретного сорта вина, страну происхождения, или другие характеристики, которые интересуют его.

– Получение результатов: После ввода запроса в поисковую строку, клиентская часть приложения отправляет запрос на сервер, который обрабатывает запрос и находит соответствующие результаты в базе данных вин. Результаты могут включать в себя информацию о вине, такую как его название, происхождение, сорт, описание, рейтинг.

– Выбор конкретного результата: После получения списка результатов пользователь может выбрать интересующий его вариант. Это может быть сделано путем нажатия на результат. Когда пользователь выбирает конкретный результат, система может предоставлять дополнительную информацию о вине.

– Навигация по страницам: Пользователь может также переходить между различными страницами, чтобы получить дополнительную информацию о вине или изучить другие варианты. Это может включать в себя переход к истории бренда, отзывам пользователей и т.д.

Таким образом, клиент-серверное приложение предоставляет пользователю возможность быстро и удобно получать информацию о винах, осуществлять поиск и выбор наиболее подходящего варианта. Это средство позволяет введение запросов, получение результатов и глубокое изучение интересующей информации о вине. Пример данной архитектуры можно увидеть на (Рис. 2.2)

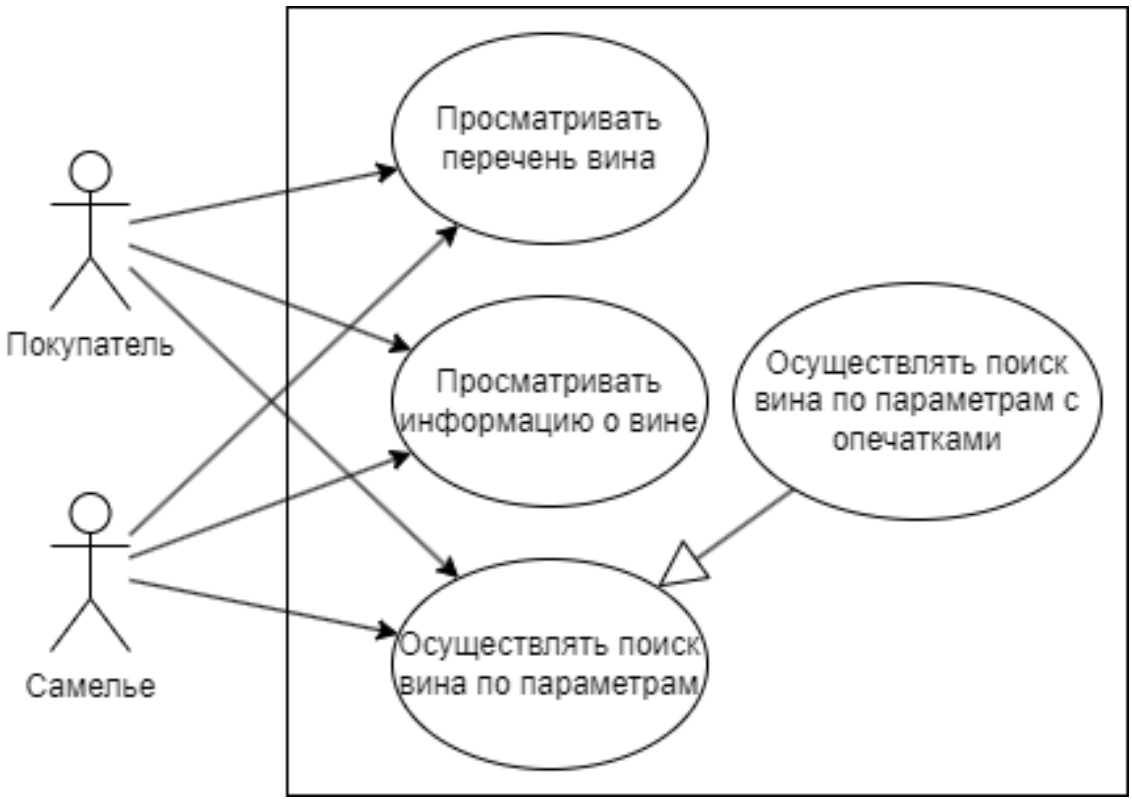


Рисунок 2.2 – Основной фукнционал клинтской части

Что касается серверной части, основной задачей её функционирования будет осуществление запросов к базе данных и получение результатов, а также применение поискового алгоритма. Обычный метод поиска не является наилучшим решением данной задачи, поскольку пользователи часто допускают ошибки при вводе данных, что приводит к неверным результатам. Для решения этой проблемы были выбраны алгоритмы, основанные на редакционном расстоянии. Из них был выбран алгоритм Левенштейна. Пример блок-схемы данного алгоритма можно найти на (Рис. 2.3)

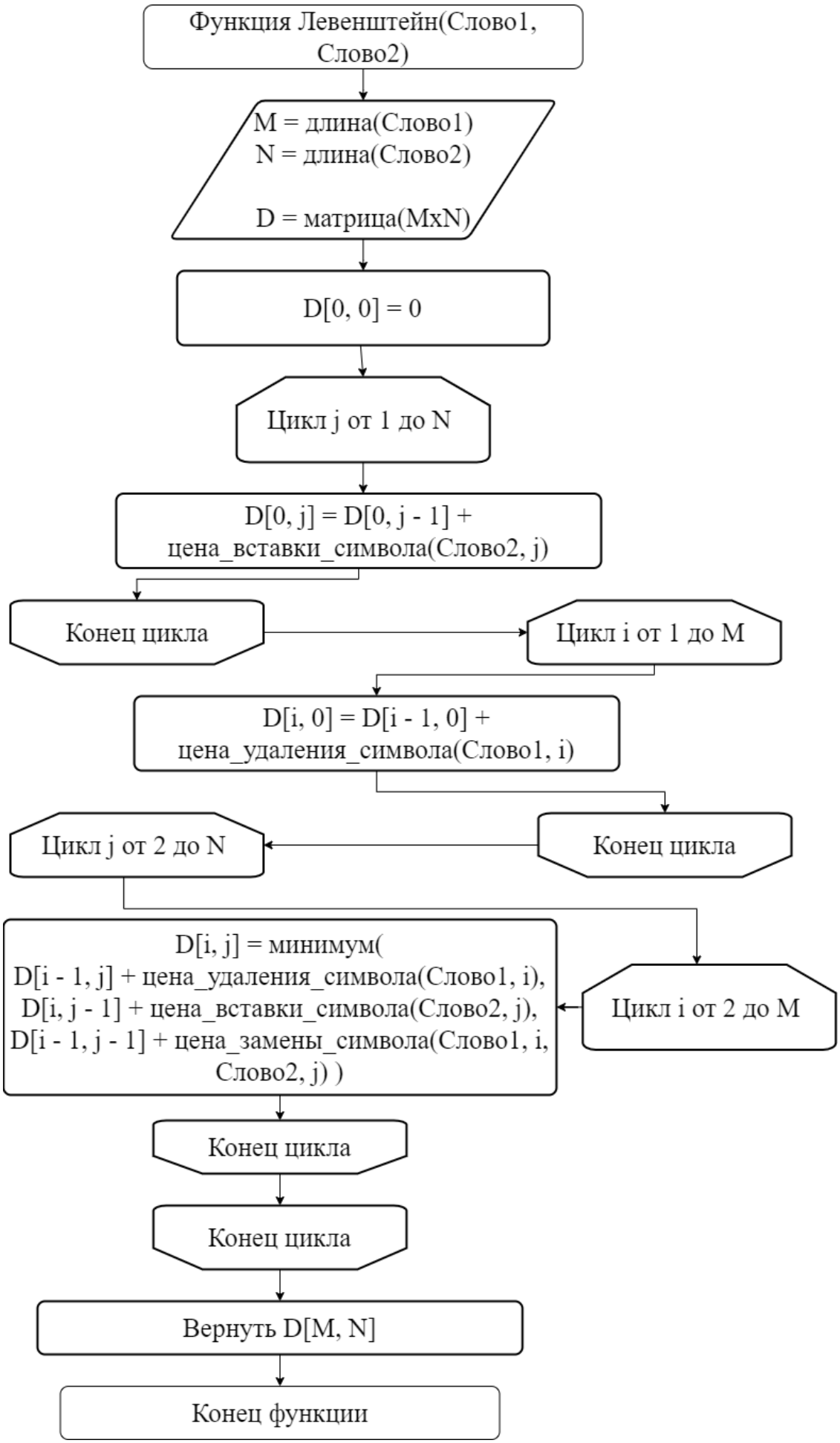


Рисунок 2.3 – Блок схема алгоритма Левенштейна

Основная идея алгоритма Левенштейна заключается в определении минимального числа операций (вставки, удаления, замены), необходимых для превращения одной строки в другую. Для этого применяется матрица размером (m+1)x(n+1), где m и n - длины соответствующих строк.

В первой строке и первом столбце матрицы записываются числа от 0 до m и от 0 до n соответственно. Затем заполняется оставшаяся часть матрицы. Каждый элемент матрицы вычисляется по формуле:

d[i,j] = min(d[i-1,j]+1, d[i,j-1]+1, d[i-1,j-1]+cost),

где d[i,j] - элемент матрицы в i-й строке и j-м столбце, cost - стоимость замены символа из i-й строки на символ из j-го столбца (равна 0, если символы равны, и 1, если не равны).

Минимальное количество операций для превращения одной строки в другую равно значению последнего элемента матрицы d[m,n]. Формульное представление этого алгоритма можно найти на (Рис. 2.3)

$$d_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{если } \min(i,j) = 0, \\ \min \begin{cases} d_{a,b}(i-1,j) + 1 \\ d_{a,b}(i,j-1) + 1 \\ d_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \\ d_{a,b}(i-2,j-2) + 1 \end{cases} & \text{если } i,j > 1 \text{ и } a_i = b_{j-1} \text{ и } a_{i-1} = b_j \\ \min \begin{cases} d_{a,b}(i-1,j) + 1 \\ d_{a,b}(i,j-1) + 1 \\ d_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{иначе,} \end{cases}$$

Рисунок 2.4 – Формула алгоритма Ливенштейна

Следует также отметить, что алгоритм Левенштейна имеет две вариации: рекурсивную и матричную. Рекурсивный метод Левенштейна может быть менее эффективным из-за повторного вычисления значений для различных подстрок. Это особенно заметно при работе с длинными строками и может привести к экспоненциальному времени выполнения. В матричном методе значения сохраняются в матрице и могут быть повторно использованы, что ускоряет процесс вычисления. В данной системе используется матричный метод для обеспечения положительного пользовательского опыта.

Для хранения необходимой информации, используется реляционная база данных MySQL. Для представления того как информация будет хранится внутри базы данных было принято решение использовать фреймовую модель. Пример того как данные будут храниться в базе данных можно увидеть на предоставленном фрейме на (Рис. 2.5). Так же была сформирована фреймовая диаграма которая показывает как данные внутри базы данных связанны между собой. Пример данной диаграммы можно увидеть на (Рис. 2.6)

Информация о вине	
PK	идентификатор_записи
	Наименование вина
	Страна_производитель
	Город_производства
	Производитель_вина
	Тип_вина
	Сорта_винограда
	Рейтинг
	Изображение_вина
	Описание_вина
	Большое_изображение

Рисунок 2.5 – Фреймовое представление хранения информации внутри MySQL

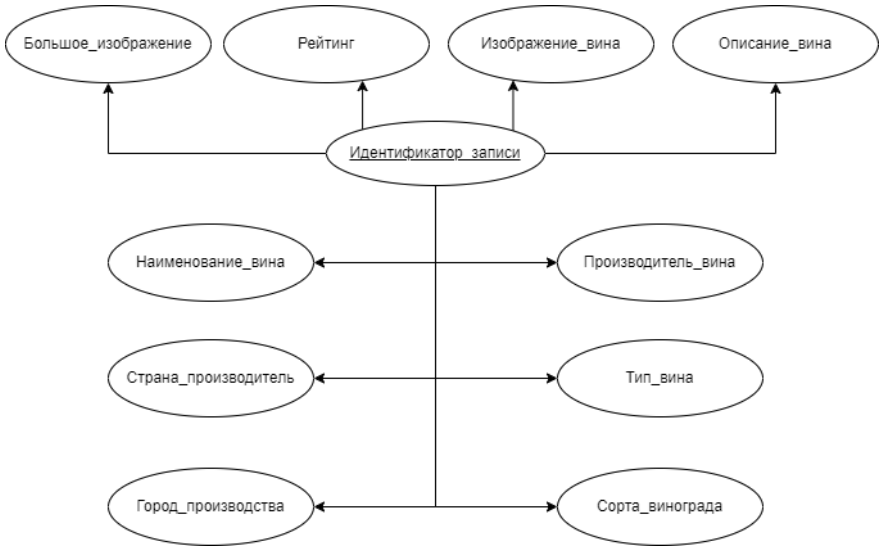


Рисунок 2.6 – Фреймовое представление хранения информации внутри MySQL

Важно подчеркнуть, что данная система ориентирована на пользователей, не имеющих предварительного опыта в данной предметной области, а также на тех, кто обладает определенным уровнем опыта, но желает получить дополнительную информацию.

2.1 Вывод

Описана предметная область, обьяснено почему данное приложение является необходимым в реализации, описанно для какой аудитории

предназначена данная предметная область. Спроектированна архитектура приложения, сформирована диаграмма USE CASE, которая показывает основной функционал клиентской части, сформирована блок схема алгоритма Левенштейна. Сформированны фреймовое предстовление

3 РАЗРАБОТКА ИНФОРМАЦИОННО СПРАВОЧНОЙ СИСТЕМЫ ПО ВИНУ

В предыдущих разделах было определено, что информационно-справочные системы, как правило, предполагают клиент-серверную реализацию. Для этого были выбраны следующие инструменты:

В начале рассматривается клиентская часть. Изначально планировалось использовать основные технологии: HTML5, CSS, JS. Однако после более глубокого изучения и анализа инструментария клиентской части было принято решение в пользу использования библиотеки для JS – React. React представляет собой JavaScript-библиотеку с открытым исходным кодом, предназначенную для разработки пользовательских интерфейсов[6]. Её разработкой и поддержкой занимаются компании Facebook, Instagram, а также сообщество независимых разработчиков и корпораций. React может применяться для создания одностраничных и мобильных приложений. Причиной выбора данной библиотеки послужило то, что разработка на стандартном наборе технологий считается уже устаревшей. В свою очередь, React позволяет создавать приложения, которые быстро загружаются и оперативно работают, обеспечивая пользователю приятный опыт.

Для серверной части приложения, был выбран Node.js и библиотека Express.js. Node.js позволяет использовать JavaScript в серверной среде, что упрощает сопровождение и ускоряет разработку. Express.js, в свою очередь, предоставляет удобный и лаконичный способ обработки HTTP-запросов и управления маршрутами.

В качестве базы данных, было принято решение воспользоваться MySQL. Эта надежная и широко используемая реляционная СУБД обеспечит надежное хранение данных и позволит эффективно управлять информацией в приложении.

Этот выбор инструментов позволит мне создать надежное, быстродействующее и масштабируемое клиент-серверное приложение, соответствующее современным требованиям веб-разработки.

В первую очередь стоит отметить серверную часть, которая написана на Express.js. Реализацию можно увидеть на (Рис. 3.1).

Когда фреймворк Express получает запрос, этот запрос передается в конвейер обработки. Конвейер состоит из набора компонентов или middleware, которые получают данные запроса и решают, как его обрабатывать. В данной реализации встраивается конвейер обработки запроса в функцию middleware. Для этого применяется метод app.use(). Функция, которая передается в app.use(), принимает три параметра: request: данные запроса, response: объект для управления ответом, next: следующая в конвейере обработки функция. В данном случае используется bodyParser.

Так как данные отправляются с помощью метода POST, то для обработки определяем функцию app.post("/ ...). Первый параметр функции - адрес, на который идет отправка - "/". Второй параметр - запрос. Третий параметр – ответ

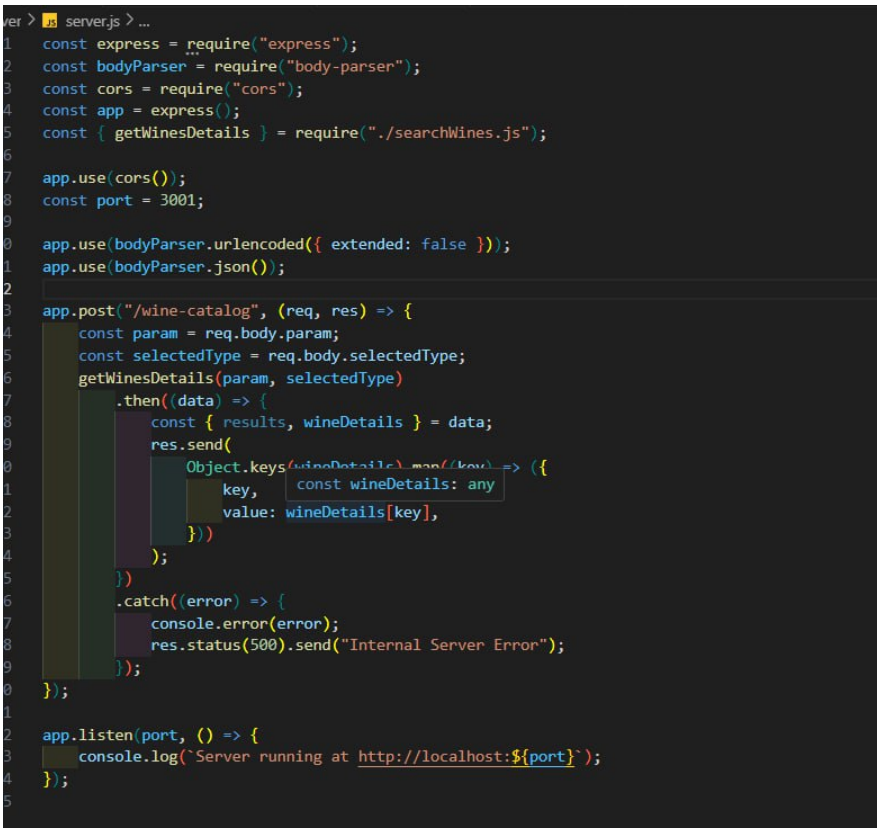


Рисунок 3.1 – Реализация серверной части

Во вторую очередь стоит отметить работу с базой данных (Рис. 3.2) . Для работы с сервером MySQL в Node.js можно использовать ряд драйверов. Самые популярные из них mysql и mysql2. По большей части они совместимы. В данном случае используется mysql2, так как, судя по ряду тестов, он предоставляет большую производительность. Для создания подключения применяется метод createConnection(), который в качестве параметра принимает настройки подключения и возвращает объект, представляющий подключение.Передаваемые в метод настройки конфигурации могут содержать ряд параметров. Наиболее используемые из них:

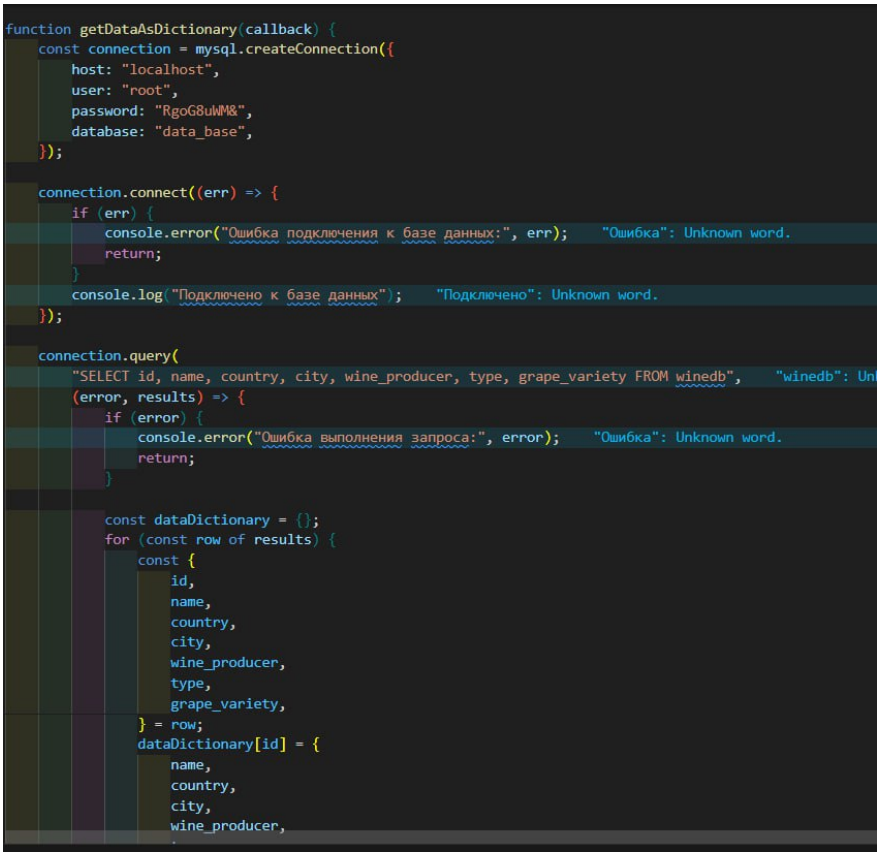
host: хост, на котором запущен сервер mysql. По умолчанию имеет значение "localhost"

user: пользователь MySQL, который используется для подключения

password: пароль для пользователя MySQL

database: имя базы данных, к которой идет подключение. Необязательный параметр. Если он не указан, то подключение идет в целом к северу

Для установки подключения мы можем использовать метод connect() объекта connection. Метод connect() принимает функцию, параметр которой содержится ошибка, которая возникла при подключении.Для выполнения запросов у объекта подключения применяется метод query().Где первый параметр - выполняемая SQL-команда, а второй - функция обратного вызова, через параметры которой мы можем получить результаты выполнения sql-команды или возникшую ошибку.

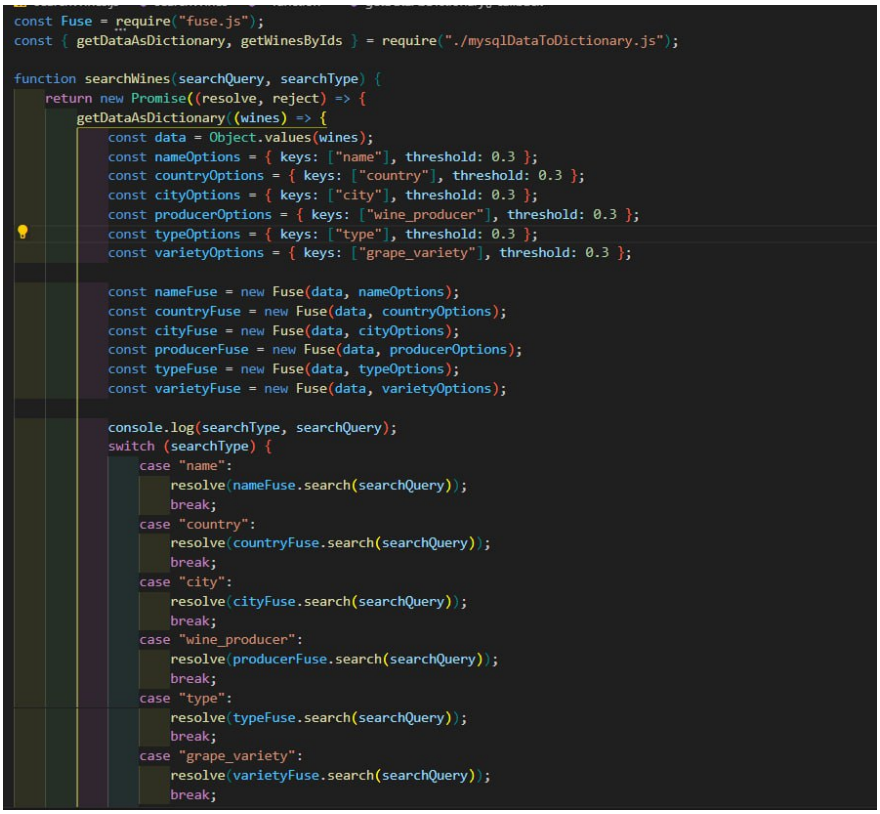


```
function getDataAsDictionary(callback) {
    const connection = mysql.createConnection({
        host: "localhost",
        user: "root",
        password: "RgoG8uWM&",
        database: "data_base",
    });

    connection.connect((err) => {
        if (err) {
            console.error("Ошибка подключения к базе данных:", err);    "Ошибка": Unknown word.
            return;
        }
        console.log("Подключено к базе данных");    "Подключено": Unknown word.
    });

    connection.query(
        "SELECT id, name, country, city, wine_producer, type, grape_variety FROM winedb",    "winedb": Un
        (error, results) => {
            if (error) {
                console.error("Ошибка выполнения запроса:", error);    "Ошибка": Unknown word.
                return;
            }

            const dataDictionary = {};
            for (const row of results) {
                const {
                    id,
                    name,
                    country,
                    city,
                    wine_producer,
                    type,
                    grape_variety,
                } = row;
                dataDictionary[id] = {
                    name,
                    country,
                    city,
                    wine_producer,
```

Рисунок 3.2 – Подключение БД и настройка

В третьих это поиск в первоначальном виде планировался алгоритм Ливенштейна, но к сожалению при тестировании данного алгоритма выдавались некорректные результаты, тогда было принято решение использовать библетеку Fuse. Данная библиотека JavaScript, предназначенная для выполнения различных операций поиска и фильтрации в массивах объектов. Она предоставляет мощные инструменты для выполнения различных типов поиска, включая размытый (fuzzy) поиск, точный поиск, поиск с использованием ранжирования и многое другое. Логика реализации следующая: В реализованном блоке, получаются все значения из базы данных и затем для каждого значения и его типа создаете отдельные объекты. Затем создается еще один объект, в который передаются данные со всеми значениями и предыдущим объектом с типом. Далее, в блоке switch, проводится поиск по конкретному заданному типу, когда приходит запрос с клиента. Пример

реализации можно увидеть на (Рис. 3.3)



```
const Fuse = require("fuse.js");
const { getDataAsDictionary, getWinesByIds } = require("./mysqlDataToDictionary.js");

function searchWines(searchQuery, searchType) {
    return new Promise((resolve, reject) => {
        getDataAsDictionary((wines) => {
            const data = Object.values(wines);
            const nameOptions = { keys: ["name"], threshold: 0.3 };
            const countryOptions = { keys: ["country"], threshold: 0.3 };
            const cityOptions = { keys: ["city"], threshold: 0.3 };
            const producerOptions = { keys: ["wine_producer"], threshold: 0.3 };
            const typeOptions = { keys: ["type"], threshold: 0.3 };
            const varietyOptions = { keys: ["grape_variety"], threshold: 0.3 };

            const nameFuse = new Fuse(data, nameOptions);
            const countryFuse = new Fuse(data, countryOptions);
            const cityFuse = new Fuse(data, cityOptions);
            const producerFuse = new Fuse(data, producerOptions);
            const typeFuse = new Fuse(data, typeOptions);
            const varietyFuse = new Fuse(data, varietyOptions);

            console.log(searchType, searchQuery);
            switch (searchType) {
                case "name":
                    resolve(nameFuse.search(searchQuery));
                    break;
                case "country":
                    resolve(countryFuse.search(searchQuery));
                    break;
                case "city":
                    resolve(cityFuse.search(searchQuery));
                    break;
                case "wine_producer":
                    resolve(producerFuse.search(searchQuery));
                    break;
                case "type":
                    resolve(typeFuse.search(searchQuery));
                    break;
                case "grape_variety":
                    resolve(varietyFuse.search(searchQuery));
                    break;
```

Рисунок 3.3 – Реализация поиска

Заключительной частью является интерфейс. Интерфейс представляется собой три страницы. Первая страница демонстрирует поисковую строку где пользователь может выбрать поиск по ключевой информации, по нажатию кнопки пользователя производится переадресация на новую страницу где выводятся результаты найденные по запросу пользователя, на данной странице так же можно продолжать производить поиск и последняя страница это страница с детальной информацией о конкретном вине, открывается она по нажатию на соответствующие вино. Все это можно увидеть на (Рис. 3.4) (Рис. 3.5) (Рис. 3.6)

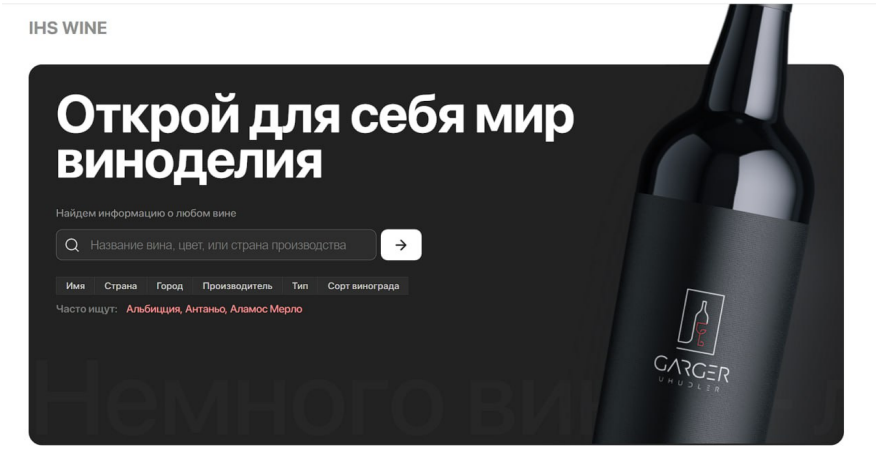


Рисунок 3.4 – Главная страница

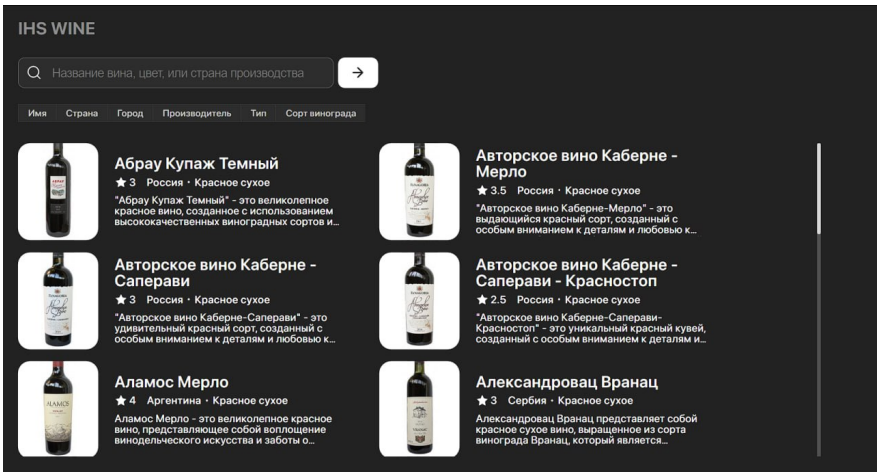


Рисунок 3.5 – Страница с результатами поиска

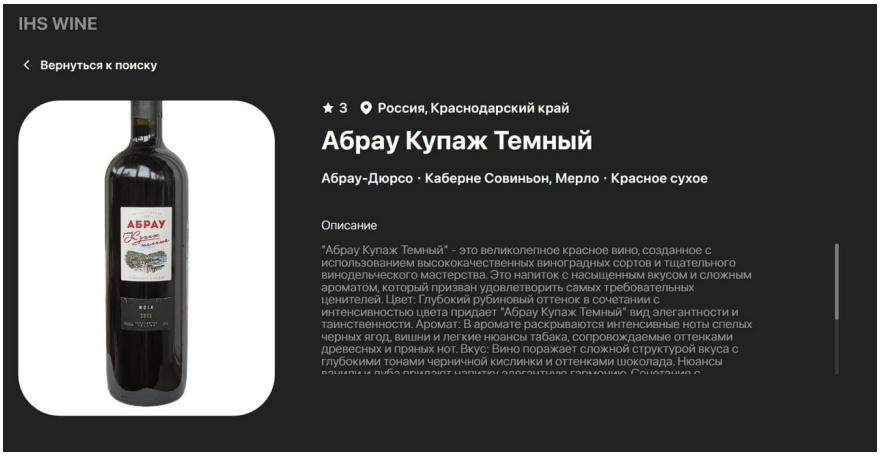


Рисунок 3.6 – Страница с конкрентным вином

3.1 Вывод

В результате курсовой работы была разработана информационно справочной системы по вину, который отвечает всем требованиям и решает

весь перечень задач, поставленных перед ним, а именно:

– Реализация Веб-приложения с архитектурой клиент-сервер

– Реализация интуитивно понятного и приятного графическикого интерфейса

– Реализация алгоритма нечеткого поиска

ЗАКЛЮЧЕНИЕ

Был проведен анализ предметной области, изучены подходы к проектированию информационно справочной системы по вину, изучены подходы к реализации информационно справочной системы по вину.Спроектирован принцип взаимодействия пользователя с сайтом, спроектирована работа сайта. Приведены основные диаграммы используемые в разработке, использованы современные средства для обеспечения приятного опыта от пользования сайта. Разработано веб-приложение удовлетворяющий всем современным требованиям

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Информационно-справочная система как средство поддержки разработки учебных дисциплин [Электронный ресурс]. — Режим доступа: https://cyberleninka.ru/article/n/informatsionno-spravochnaya-sistema-kak-sredstvo-podderzhki-razrabotki-uchebnyh-dis – Дата доступа: 02.10.2023.

[2] Веб-приложение [Электронный ресурс]. — Режим доступа: https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5. – Дата доступа: 02.10.2023.

[3] Разработка веб-приложений: языки и инструменты [Электронный ресурс]. — Режим доступа: https://ya.zerocoder.ru/pgt-razrabotka-veb-prilozhenij-yazyki-i-instrumenty/. – Дата доступа: 02.10.2023.

[4] Веб-фреймворк [Электронный ресурс]. — Режим доступа: https://appmaster.io/ru/glossary/veb-freimvork. – Дата доступа: 02.10.2023.

[5] 10 Лучших Фреймворков для Фронтенда и Бэкенда [Электронный ресурс]. — Режим доступа: https://blog.back4app.com/ru/10-%D0%BB%D1%83%D1%87%D1%88%D0%B8%D1%85-%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA%D0%BE%D0%B2-%D0%B4%D0%BB%D1%8F-%D1%84%D1%80%D0%BE%D0%BD%D1%82%D0%B5%D0%BD%D0%B4%D0%B0-%D0%B8-%D0%B1%D1%8D/#:~:text=%D0%91%D1%8D%D0%BA%D0%B5%D0%BD%D0%B4%20%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA%D0%B8%20%E2%80%93%20%D1%8D%D1%82%D0%BE%20%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B8%20%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D1%8B%D1%85,%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B8%20%D0%B2%D0%BD%D1%83%D1%82%D1%80%D0%B5%D0%BD%D0%BD%D0%B5%D0%B9%20%D0%BA%D0%BE%D0%BD%D1%84%D0%B8%D0%B3%D1%83%D1%80%D0%B0%D1%86%D0%B8%D0%B8%20%D0%B2%D0%B5%D0%B1%2D%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9.. – Дата доступа: 02.10.2023.

[6] Что такое ReactJS? [Электронный ресурс]. — Режим доступа: http://web.spt42.ru/index.php/chto-takoe-reactjs. – Дата доступа: 02.10.2023.

[7] Введение в серверную часть [Электронный ресурс]. — Режим доступа: https://developer.mozilla.org/ru/docs/Learn/Server-side/First_steps/Introduction. – Дата доступа: 02.10.2023.

[8] Что такое база данных? [Электронный ресурс]. — Режим доступа: https://www.oracle.com/cis/database/what-is-database/. – Дата доступа: 02.10.2023.