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

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки / специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Программная реализация алгоритмов синтеза расписаний обслуживания

Обучающийся В.В. Попрас
(Инициалы Фамилия) (личная подпись)

Руководитель М.А. Тренина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант Н.В. Ященко
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

Данная выпускная квалификационная работа направлена на реализацию программного алгоритма для комплексного планирования услуг. В данной работе рассматриваются задачи оптимизированного планирования для различных видов деятельности, включая транспорт, медицину, образование и производство. Целью данного исследования является разработка программного обеспечения для автоматизированных протоколов синтеза на основе современных алгоритмов оптимизации.

Проанализированы существующие подходы к планированию с учетом эвристических и точных алгоритмов. В ходе данного исследования было разработано, протестировано и оценено на эффективность программное обеспечение, реализующее выбранные алгоритмы.

Методы исследования включают теоретический анализ научной литературы, математическое моделирование и программную реализацию алгоритмов. Разработанный метод обладает высокой гибкостью и адаптируется к терминологии различных приложений, что позволяет использовать его для автоматизации процесса диспетчеризации.

Работа состоит из трех глав, включающих 59 страниц, 20 рисунков, 12 таблиц и 26 источника литературы. Область применения результатов включают автоматизацию планирования и управления ресурсами в различных областях. Новизна данной работы заключается в комбинированной адаптации и использовании современных алгоритмов оптимизации, что позволяет значительно повысить эффективность планирования времени. Эти результаты могут быть применены в практической деятельности эффективных организаций по планированию ресурсов. Внедрение разработанных решений повышает производительность, сокращает временные затраты и сводит к минимуму конфликты планирования.

Abstract

This final qualification work is aimed at the implementation of a software algorithm for integrated service planning. This work considers the problems of optimized planning for various activities, including transport, medicine, education and production. The purpose of this study is to develop software for automated synthesis protocols based on modern optimization algorithms.

Existing approaches to planning are analyzed taking into account heuristic and exact algorithms. In the course of this study, software implementing the selected algorithms was developed, tested and evaluated for efficiency.

Research methods include theoretical analysis of scientific literature, mathematical modeling and software implementation of algorithms. The developed method is highly flexible and adapts to the terminology of various applications, which allows it to be used to automate the dispatching process.

The work consists of three chapters, including 59 pages, 20 figures, 12 tables and 26 sources of literature. The scope of application of the results includes automation of planning and resource management in various fields. The novelty of this work lies in the combined adaptation and use of modern optimization algorithms, which allows to significantly increase the efficiency of time planning. These results can be applied in the practical activities of effective organizations in resource planning. The implementation of the developed solutions increases productivity, reduces time costs and minimizes planning conflicts.

Оглавление

Введение

Современные системы управления ресурсами требуют эффективного планирования технического обслуживания в различных сферах, таких как транспорт, медицина, образование, производство и т. д. Планирование оптимизации минимизирует затраты, повышает эффективность использования персонала и оборудования, а также улучшает качество предоставляемых услуг. Однако традиционные методы создания плана требуют больше времени и не всегда дают наилучшие результаты. В этом контексте соответствующей задачей является автоматизация процесса планирования с использованием комплексных алгоритмов.

Автоматизированные системы синтеза решений делают процесс более гибким и адаптивным, учитывая множество параметров и ограничений. Для поиска наиболее эффективного решения можно использовать передовые методы и алгоритмы оптимизации, такие как генетические алгоритмы, роевые методы и эвристики. Разработка программного обеспечения для автоматизации процесса синтеза расписаний является важнейшей задачей, которая может повысить производительность и сократить объем операций.

Целью данной работы является. разработка программного обеспечения для решения задачи интеграции планирования услуг на основе современных алгоритмов оптимизации

Для достижения этой цели необходимо решить следующие задачи:

- анализ существующих методов и алгоритмов синтеза планов;
- упростить оптимальный алгоритм для реализации в программном обеспечении;
- разработать архитектуру программы, обеспечивающую гибкость и эффективность:
- реализовать выбранные методы и алгоритмы для интеграции планов программного кода;

– тестирование и оценка эффективности разработанного программного обеспечения.

Объект исследования: процесс разработки плана технического обслуживания. Предмет исследования: программные методы и алгоритмы для планирования синтеза.

В работе использованы следующие методы исследования:

– теоретический анализ научно–технической литературы по интеграции алгоритмов планирования и оптимизации;
– математическое моделирование для формализации задачи и выбора оптимальных подходов;
– программная реализация и практическое тестирование разработанных алгоритмов.

В работу включен список реализаций, ключевых моментов, выводов и ссылок. В главе 1 рассматриваются теоретические основы комплексного планирования и анализ существующих методов и алгоритмов. В главе 2 обсуждается процесс разработки программного обеспечения, обоснование выбора алгоритма и его реализация. Глава 3 посвящена тестированию и оценке эффективности разработанного решения. Таким образом, можно подвести итоги работы и принять наиболее важные решения.

Глава 1 Теоретические основы синтеза расписаний

1.1 Понятие расписания и задачи синтеза расписаний

Расписание представляет собой набор процедур для немедленного выполнения определенных действий или событий. Расписание используется в различных сферах деятельности, таких как образование, транспорт, производство, медицина и логистика. Основная цель создания плана – оптимально распределить ресурсы (время, оборудование, людей) в соответствии с различными ограничениями и требованиями.

Синтез расписание – это процесс автоматизированного или ручного планирования, отвечающий определенным критериям надежности [18]. Основными целями планируемой интеграции являются:

- минимизация затрат на реализацию процесса планирования;
- максимально использование имеющихся ресурсов;
- учет различных ограничений, таких как смена работы, навыки работы с оборудованием, изменяемые стандарты работы, гибкость.

В работе обсуждается интеграция планирования услуг с точки зрения реализации алгоритмического программного обеспечения. Для решения можно использовать различные подходы: эвристические алгоритмы, методы линейного программирования, комбинаторные алгоритмы и машинное обучение [21]. Выбор метода зависит от сложности задачи, количества входных параметров, требований к точности и надежности решения.

1.2 Описание математической модели

Математическая модель синтеза расписаний представляет собой формализованное описание задачи планирования, включающее множество переменных, ограничений и целевых функций [15]. Рассмотрим основные компоненты модели.

Пусть:

$T = \{t_1, t_2, \dots, t_n\}$ – множество временных интервалов.

$R = \{r_1, r_2, \dots, r_m\}$ – множество доступных ресурсов.

$A = \{a_1, a_2, \dots, a_k\}$– множество задач, которые необходимо решить.

$x_{(i,j)}$ – бинарная переменная, принимающая значение 1, если задача a_i выполняется в интервале t_j и 0 в противном случае.

Задача оптимизации расписания может включать разные целевые функции:

– минимизация общего времени выполнения (1):

$$min \sum_{i=1}^{k} \sum_{j=1}^{n} t_j x_{i,j}, \quad (1)$$

– максимальное использование ресурсов (2):

$$min \sum_{i=1}^{k} \sum_{j=1}^{n} r_j x_{i,j}. \quad (2)$$

Существуют различные ограничения. Ограничение на выполнение задачи в один момент времени (3):

$$\sum_{j=1}^{n} x_{i,j} = 1, \ \ \forall i \in A. \quad (3)$$

Это условие гарантирует, что каждая задача выполняется ровно один раз. Ограничение на ресурсы (4):

$$\sum_{i=1}^{k} x_{i,j} \leq r_j, \forall_j \in T. \quad (4)$$

Это ограничение обеспечивает выполнение задач в рамках доступных ресурсов. Последовательность выполнения (5):

$$t_{j_1} x_{i,j_2} \leq t_{j_1} x_{i,j_2}, \forall_i \in A, j_1 < j_2. \quad (5)$$

Это ограничение необходимо для задач, требующих строгого порядка выполнения.

Предложенная модель позволяет гибко управлять процессом синтеза расписаний, минимизируя время выполнения и обеспечивая оптимальное распределение ресурсов. Выбранные методы решения включают линейное программирование и жадные алгоритмы, в зависимости от сложности входных данных.

1.3 Классификация методов синтеза расписаний

1.3.1 Детерминированные методы

Методы синтеза расписаний можно классифицировать по различным критериям, таким как степень детерминированности, адаптивность и вычислительная сложность. Рассмотрим детерминированные методы синтеза расписаний.

Детерминированные методы синтеза расписаний представляют собой подходы, при которых результат полностью определяется исходными данными и алгоритмом. Эти методы дают одно и то же решение при каждом запуске с одинаковыми входными данными. Ключевые особенности детерминированных методов – точность, предсказуемость и возможность строгого математического обоснования.

Метод полного перебора (Brute Force) заключается в рассмотрении всех возможных вариантов расписания и выборе наилучшего по заданным критериям.

Формально, если имеется n операций, которые могут выполняться на m ресурсах, количество возможных расписаний определяется как (6):

$$S=P(n,m)=n!/(n-m)!, \quad (6)$$

где P(n,m) – число размещений n элементов по m позициям.

Этот метод гарантирует нахождение оптимального расписания, но имеет экспоненциальную сложность O(n!), что делает его непрактичным для больших задач.

Жадные алгоритмы (Greedy Algorithms) строят расписание, последовательно выбирая наилучший локальный вариант на каждом шаге. Такой подход не всегда приводит к глобально оптимальному решению, но обеспечивает хорошую скорость работы.

Жадный алгоритм для задачи планирования выполнения n задач:

– отсортировать задачи по возрастанию времени выполнения t_i;

– назначить первую задачу на свободное время;

– повторять, пока все задачи не будут распределены.

Жадный критерий (7):

$$min \sum_{i=1}^{n} C_i, \quad (7)$$

где C_i – момент завершения i–й задачи.

Метод критического пути (Critical Path Method, CPM) применяется в проектном управлении и задачах с зависимостями между операциями.

Для каждой задачи определяется время выполнения t_i и зависимости между задачами.

Строится ориентированный граф G=(V,E), где вершины V – задачи, а рёбра E обозначают зависимости.

Вычисляется длина критического пути (8):

$$\text{CP} = max \sum_{i \in \text{P}} t_i, \quad (8)$$

где P – последовательность задач на критическом пути.

Этот метод позволяет определить минимальное время выполнения всех операций.

Метод динамического программирования (Dynamic Programming) применяется, если задачу можно разбить на подзадачи, решаемые рекурсивно с запоминанием результатов.

Пример для задачи о разбиении расписания с минимальными затратами (9):

$$C(i,j) = \min_k \{C(i,k) + C(k+1,j) + d(i,j)\}, \quad (9)$$

где $C(i,j)$ – стоимость расписания от операции i до j,

$d(i,j)$ – дополнительное ограничение.

Сложность метода – полиномиальная $O(n^2)$, что делает его эффективнее полного перебора.

Детерминированные методы находят применение в задачах логистики, планирования производства, организации работы транспортных систем и других областях, где важно строгое соответствие расписания заданным параметрам.

1.3.2 Стохастические методы

Стохастические методы синтеза расписаний опираются на вероятностные модели и используются в условиях неопределенности. Эти методы применяются в случаях, когда исходные данные могут изменяться во времени или содержать случайные величины.

Генетические алгоритмы (Genetic Algorithms, GA) имитируют принципы эволюции и естественного отбора. Решения (индивиды) представляются в виде хромосом, которые эволюционируют с помощью операторов мутации и скрещивания.

1. Инициализация: создаётся начальная популяция случайных решений.

2. Вычисление функции приспособленности (fitness function) (10):

$$F(S) = \sum_{i=1}^{n} C_i w_i, \quad (10)$$

где C_i – момент завершения i–й задачи,

w_i – её вес.

3. Отбор лучших особей.

4. Операции скрещивания и мутации для создания новой популяции.

5. Повторение до достижения критерия останова (например, фиксированное количество итераций или отсутствие улучшений).

Этот метод особенно эффективен при сложных задачах, где число возможных расписаний слишком велико для перебора.

Метод имитации отжига основан на аналогии с физическим процессом кристаллизации металлов.

1. Начальное решение S выбирается случайно.

2. Вычисление целевой функция $F(S)$.

3. Генерация нового решение $F(S')$ путём небольшого изменения.

4.1 Если $F(S')$ лучше, принимаем его.

4.2 Если хуже, принимаем его с вероятностью (11):

$$P = e^{\frac{-F(S')-F(S)}{T}}, \quad (11)$$

где T – температура, постепенно уменьшающаяся по закону (12):

$$T = T_0 \cdot \alpha k, 0 < \alpha < 1. \quad (12)$$

Метод позволяет избежать локальных минимумов за счёт принятия временно худших решений.

Муравьиные алгоритмы моделируют поведение колонии муравьёв, которые оставляют феромонный след при поиске кратчайшего пути.

Каждое расписание представляется в виде пути графа.

Муравьи случайно перемещаются по узлам, оставляя феромоны (13):

$$\tau_{i,j}(t+1) = (1-\rho)\tau_{ij}(y) + \sum_{k} \Delta\,\tau_{ij}^{k}, \qquad (13)$$

где ρ – коэффициент испарения,

$\Delta\tau_{ij}^{k}$ – обновление феромонов.

Алгоритм эффективен при решении комбинаторных задач, таких как синтез расписаний.

Стохастические методы позволяют находить качественные расписания даже для сложных задач, однако они не всегда гарантируют точное оптимальное решение, а их эффективность зависит от параметров настройки.

Стохастические методы широко применяются в задачах распределения ресурсов, управления проектами, логистике и медицинском планировании, где важно учитывать неопределенность внешних факторов.

1.3.3 Гибридный подходы

Гибридные методы синтеза расписаний комбинируют элементы детерминированных и стохастических методов для достижения более эффективных решений. Они позволяют учитывать, как строгие ограничения, так и неопределенность внешних факторов, обеспечивая баланс между точностью и адаптивностью.

Гибридные методы комбинируют элементы различных классов алгоритмов для достижения наилучшего качества расписания. Обычно это сочетание детерминированных и стохастических методов, позволяющее объединить точность первых с гибкостью вторых.

Гибридизация генетических алгоритмов и локального поиска (GA + Local Search)

Один из популярных гибридных методов – улучшение генетического алгоритма с помощью локального поиска.

Генетический алгоритм создаёт начальную популяцию и выполняет основные операции (скрещивание, мутация).

После каждого поколения лучшие особи дополнительно оптимизируются с помощью локального поиска (например, метода градиентного спуска).

Функция приспособленности рассчитывается как (14):

$$F(S) = \sum_{i=1}^{n} C_i w_i, \tag{14}$$

где C_i – время завершения i–й операции,

w_i – её вес.

Этот метод позволяет повысить точность решений за счёт детальной доработки кандидатов.

Имитация отжига + муравьиные алгоритмы (SA + ACO). Метод сочетает способность муравьиных алгоритмов находить глобально хорошие решения с возможностью имитации отжига избегать локальных минимумов. Такой метод даёт баланс между случайным поиском и локальной оптимизацией.

Гибридизация детерминированных и эвристических методов (Mixed Integer Programming + Heuristics). В этом подходе основа расписания формируется с помощью метода целочисленного линейного программирования (MIP), а затем корректируется эвристическими методами. Затем используется жадный алгоритм или генетический метод для улучшения решения.

Гибридные методы позволяют добиться высокой эффективности и применяются в сложных задачах, таких как управление производственными процессами или расписаниями авиаперелётов.

Гибридные методы находят применение в логистике, управлении производственными процессами и планировании в динамически изменяющихся системах.

1.4 Анализ существующих подходов к решению задач синтеза расписаний

Анализ существующих методов синтеза расписаний позволяет определить их сильные и слабые стороны, а также применимость в различных условиях (таблица 1).

Таблица 1 – Существующие подходы

Подход	Описание	Преимущества	Недостатки	Примеры применения
Математическое программирование	Линейное и целочисленное программирование для оптимизации расписаний.	Гарантированное нахождение оптимального решения, высокая точность.	Высокая вычислительная сложность, требует значительных ресурсов.	Оптимизация расписаний транспортных систем, планирование производства.
Эвристические алгоритмы	Основаны на эмпирических правилах и приближённых методах - жадные алгоритмы, метод роя частиц и генетические алгоритмы.	Быстродействие, возможность работы в реальном времени.	Отсутствие гарантии оптимального решения, необходимость подбора параметров.	Управление расписанием авиарейсов, распределение производственных ресурсов.
Методы искусственного интеллекта	Используют машинное обучение, нейросети и методы глубокого обучения для предсказания и оптимизации расписаний.	Адаптивность, возможность самообучения и работы в динамической среде.	Требуют большого объёма данных для обучения, сложность реализации.	Автоматизированные системы управления производством, интеллектуальные транспортные системы.

1.5 Критерии оценки качества расписаний

1.5.1 Временные критерии (время выполнения всех операций)

Оценка качества расписаний включает анализ различных критериев, среди которых временные показатели играют ключевую роль. Основными временными критериями являются общее время выполнения расписания C_{max} – максимальное время завершение всех операций в расписании (15):

$$C_{max} = \max_i C_i, \tag{15}$$

где C_i – время завершения i–й операции.

Среднее время выполнения задач C_{extavg} – среднее время завершения всех операций (16):

$$C_{extavg} = \frac{1}{n}\sum_{i=1}^{n} C_i C_{extavg}, \tag{16}$$

где n – общее количество операций.

Время простоя ресурсов – суммарное время, в течение которого ресурсы не задействованы (17):

$$I = C_{max} - \sum_{j=1}^{m} T_j, \tag{17}$$

где T_j – время работы j–го ресурса,

m – количество ресурсов.

Среднее время ожидания W – средняя задержка перед выполнением операций (18):

$$W = \frac{1}{n}\sum_{i=1}^{n}(S_i - A_i), \quad (18)$$

где S_i – время начала i–й операции,

A_i – момент её поступления в систему.

Применение данных критериев позволяет проводить сравнительный анализ различных подходов к синтезу расписаний и выбирать наиболее эффективные решения.

1.5.2 Ресурсные ограничения (ограничения по ресурсам)

Ресурсные ограничения являются важным фактором при синтезе расписаний, поскольку они определяют доступность и использование различных ресурсов в системе. Основные виды ресурсных ограничений:

- ограничение по количеству ресурсов – ограничение на общее число доступных единиц ресурса в определенный момент времени;
- временные ограничения ресурсов – определение временных окон, в течение которых ресурс может использоваться;
- последовательность использования ресурсов – требование, чтобы определенные ресурсы использовались в строгом порядке;
- эксклюзивность использования – условие, при котором ресурс может быть занят только одной операцией в заданный момент времени.

Примеры ресурсных ограничений представлены в таблице 2.

Таблица 2 – Ресурсные ограничения

Тип ограничения	Описание	Пример
Ограничение по количеству ресурсов	Доступно ограниченное количество ресурсов	В производственном процессе имеется только 3 станка для обработки деталей
Временные ограничения ресурсов	Ресурсы доступны только в определенное время	Рабочий персонал доступен с 8:00 до 17:00

Продолжение таблицы 2

Тип ограничения	Описание	Пример
Последовательность использования ресурсов	Определенные операции требуют использования ресурсов в заданном порядке	В сборочном процессе детали должны сначала свариваться, а затем окрашиваться
Эксклюзивность использования	Ресурс может быть занят только одной задачей в определенный момент времени	Один кран может поднимать только один груз за раз

Учёт этих ограничений при разработке расписания позволяет повысить эффективность использования ресурсов и избежать конфликтов в процессе выполнения операций.

1.5.3 Критерий минимизации затрат

Минимизация затрат при составлении расписаний является одной из ключевых целей оптимизации. Основные затраты могут включать:

– операционные затраты C_0 – затраты на выполнение задач (например, заработная плата, амортизация оборудования);

– затраты на простои C_d – затраты;

– штрафные санкции C_p – затраты за несоблюдение сроков.

Общая функция стоимости расписания может быть представлена в следующем виде (19):

$$C_{total} = C_0 + C_d + C_p. \quad (19)$$

Функция операционных затрат (20):

$$C_0 = \sum_{i=1}^{n} c_i t_i, \quad (20)$$

где c_i – стоимость работы i–го ресурса в единицу времени,

t_i – время использования i–го ресурса.

Функция затрат на простои (21):

$$C_d = \sum_{j=1}^{m} d_j T_j, \tag{21}$$

где d_j – стоимость простоя j–го ресурса в единицу времени,

T_j – общее время простоя j–го ресурса.

Функция штрафных санкций (22):

$$C_p = \sum_{k=1}^{p} p_k D_k, \tag{22}$$

где p_k – штраф за задержку k–й операции,

D_k – время задержки выполнения k–й операции.

Оптимальное расписание должно минимизировать C_{total}, обеспечивая сбалансированное использование ресурсов и минимальные затраты на выполнение операций.

Выводы по главе 1

В данной главе рассмотрены теоретические основы синтеза расписаний, определено понятие расписания и задачи, связанные с его построением. Проанализированы основные математические модели, используемые для формализации задачи синтеза, приведена классификация методов: детерминированные, стохастические и гибридные подходы.

Рассмотрены подходы к решению задач синтеза расписаний, преимущества и ограничения. Особое внимание уделено критериям оценки качества расписаний: временные показатели, ресурсные ограничения и минимизацию затрат. Проведенный анализ позволит обоснованно выбрать методологию для разработки программной реализации алгоритмов синтеза расписаний в последующих главах.

Глава 2 Алгоритмизация процесса синтеза расписаний

2.1 Выбор алгоритма для реализации

2.1.1 Основания для выбора конкретного алгоритма

Разработка плана технического обслуживания оборудования требует использования алгоритмов. Это обеспечивает оптимальное распределение ресурсов в контролируемых условиях. В данной работе для достижения этих целей используется линейное программирование, обеспечивающее оптимальное решение с определенными ограничениями. При выборе подхода линейного программирования необходимо учитывать несколько важных моментов. Входные данные содержат множество параметров с различными параметрами, такими как длительность, частота, ресурсы и приоритет. Это задача многоресурсного планирования, требующая учета человеческих ресурсов, материально–технической базы и других факторов.

Линейное программирование позволяет создавать графики, которые минимизируют простои оборудования и равномерно распределяют нагрузку между вашими сотрудниками. Кроме того, этот метод можно использовать для решения задач оптимизации. Минимизировать продолжительность работы за счет равномерного распределение нагрузки в течение всего времени.

Гибкость этого метода делает его пригодным для расширения, позволяя системе изменять конфигурацию оборудования, обновлять методы обслуживания и обновлять различные доступные ресурсы. В следующем разделе более подробно описывается, как использовать линейное программирование для планирования технического обслуживания оборудования.

2.1.2 Описание математической модели выбранного алгоритма

Для формирования расписания технического обслуживания оборудования используется метод линейного программирования. Модель учитывает множество факторов, таких как периоды технического обслуживания, доступность ресурсов и приоритетность задач.

Пусть $x_{i,j}$ – бинарная переменная, где $x_{i,j} = 1$, если задача i выполняется в временной точке j, и $x_{i,j} = 0$ в противном случае. Обозначим:

n – количество заданий (мероприятий по ТО);

m – количество доступных временных слотов;

T_i – длительность выполнения задания i;

R_k – количество доступных ресурсов типа k;

$C_{i,k}$ – количество ресурсов типа k, необходимых для выполнения задания i.

Каждое задание должно быть запланировано ровно один раз (23):

$$\sum_{j=1}^{m} x_{i,j} = 1, \forall_i \in \{1, \dots, n\}. \quad (23)$$

Выполнение задания возможно только в пределах доступного периода (24):

$$\sum_{j=t_i}^{t_i+T_i} x_{i,j}\,, \forall_i. \quad (24)$$

Ограничения по ресурсам (25):

$$\sum_{i=1}^{n} x C_{i,k} x_{i,j} \leq R, \forall k, \forall_j. \quad (25)$$

В качестве целевой функции может использоваться минимизация общего времени завершения работ (26):

$$min \sum_{i=1}^{n} \sum_{j=1}^{m} j \cdot x_{i,j}, \quad (26)$$

или минимизация нагрузки на ресурсы (27):

$$min \max_{j} \sum_{i=1}^{n} x_{i,j}. \quad (27)$$

Данная модель позволяет сформировать расписание технического обслуживания оборудования с учетом ограничений по времени и ресурсам, а также оптимизировать его по выбранному критерию [17, 22].

Вот практический пример расчета расписания технического обслуживания оборудования на основе линейного программирования.

Рассмотрим три единицы оборудования, требующих обслуживания. Каждое задание имеет свою продолжительность, приоритет и требования к ресурсам. Временной горизонт расписания составляет 6 временных слотов (таблица 3).

Таблица 3 – Перечень оборудования

Задание (i)	Длительность (T_i)	Периодичность (интервал)	Требуемые ресурсы (кол–во специалистов)	Приоритет
ТО–1	2 часа	6 дней	2	Высокий
ТО–2	3 часа	3 дня	1	Средний
ТО–3	1 час	2 дня	1	Низкий

Максимально доступное количество специалистов в каждый временной слот 3 (таблица 4).

Таблица 4 – Оптимизированное расписание

Временной слот (j)	ТО–1 (x_{1j})	ТО–2 (x_{2j})	ТО–3 (x_{3j})	Используемые ресурсы
1	1	0	1	3 (2+0+1)
2	1	0	0	2 (2+0+0)
3	0	1	0	1 (0+1+0)
4	0	1	0	1 (0+1+0)
5	0	1	0	1 (0+1+0)
6	0	0	1	1 (0+0+1)

Составленное расписание обеспечивает выполнение всех технических обслуживаний в установленные сроки, с учетом ресурсных ограничений. Высокоприоритетное ТО–1 запланировано в первые два временных слота. ТО–2 распределено равномерно по доступным ресурсам, а низкоприоритетное ТО–3 выполняется в моменты, когда есть свободные специалисты.

2.2 Проектирование алгоритмической части программы

2.2.1 Пошаговая схема работы алгоритма

Разработка алгоритма синтеза расписаний технического обслуживания оборудования требует детального проектирования, включающего формализацию входных данных, выбор метода оптимизации и реализацию вычислительного процесса. Основной целью является создание алгоритма, который автоматически формирует расписание, учитывая технические требования, регламентированную периодичность, доступность ресурсов и заданные критерии оптимизации.

Алгоритм синтеза расписаний основан на методах линейного программирования и включает следующие основные этапы (рисунок 1):

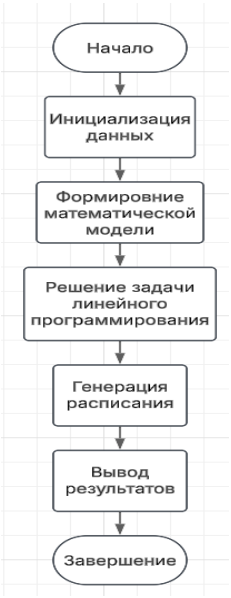

Рисунок 1 – Алгоритм работы

На первом этапе происходит инициализация данных, включающая загрузку сведений об оборудовании, таких как перечень единиц, их технические параметры и текущие эксплуатационные данные. Определяются требования к техническому обслуживанию, включая периодичность работ, нормативные требования и критические сроки. Учитываются доступные ресурсы, в том числе квалифицированные специалисты, инструменты и временные ограничения.

Далее формируется математическая модель. Определяются переменные, включая бинарные переменные для назначения ТО в конкретный временной слот и переменные для учета доступности ресурсов. Формулируются ограничения, обеспечивающие, что каждое задание запланировано ровно один раз, исключены конфликты по времени и ресурсам, а также соблюдена периодичность проведения ТО. Определяется целевая функция, например, минимизация общего времени выполнения ТО или равномерное распределение нагрузки.

На следующем этапе происходит решение задачи линейного программирования. Запускается алгоритм оптимизации, такой как симплекс–

метод или метод ветвей и границ, после чего вычисляется расписание, соответствующее заданным ограничениям и минимизирующее критерий оптимальности.

После этого выполняется генерация расписания, где оптимизированные временные интервалы назначаются соответствующим заданиям. Проверяется корректность решения на отсутствие конфликтов и соблюдение регламентов. В случае невозможности удовлетворить все ограничения осуществляется корректировка входных параметров или релаксация условий.

На завершающем этапе выполняется вывод результатов. Генерируется итоговое расписание ТО, рассчитываются дополнительные метрики, такие как коэффициент загрузки ресурсов и среднее время простоя оборудования. Пользователю предоставляется возможность корректировки и обновления данных. В результате работы алгоритма формируется оптимизированное расписание, которое снижает затраты на обслуживание и минимизирует простои оборудования.

2.2.2 Учет ограничений и особенностей задачи

При составлении расписания технического обслуживания оборудования необходимо учитывать ряд ограничений и особенностей, связанных с ресурсами, периодичностью выполнения работ и приоритетами. Ограничения определяют допустимые варианты планирования и влияют на выбор оптимального решения (таблица 5).

Таблица 5 – Ограничения и их учет в модели

Ограничение	Описание	Способ учета в модели
Доступность ресурсов	В каждый момент времени число занятых специалистов не может превышать доступного количества.	Вводится ограничение, контролирующее суммарное использование ресурсов в каждый временной слот.

Продолжение таблицы 5

Ограничение	Описание	Способ учета в модели
Периодичность ТО	Каждое оборудование должно проходить техническое обслуживание в установленные сроки.	Добавляются интервальные ограничения, гарантирующие соблюдение заданных регламентов.
Временные окна	Обслуживание может выполняться только в рабочие часы или в заранее определенные временные интервалы.	Вводятся бинарные переменные, исключающие возможность планирования вне допустимых временных окон.
Приоритет заданий	Некоторые задачи имеют более высокий приоритет и должны выполняться в первую очередь.	Используется весовая функция, обеспечивающая выполнение приоритетных заданий раньше менее значимых.
Длительность работ	Операции ТО требуют определенного времени для завершения.	Временные ограничения обеспечивают непрерывность выполнения каждой задачи.
Совместимость оборудования	Некоторые работы требуют наличия конкретных инструментов или выполнения в определенной последовательности.	Вводятся логические ограничения, учитывающие совместимость и порядок выполнения операций.

Рассмотрим три единицы оборудования с различными требованиями к техническому обслуживанию (таблица 6).

Таблица 6 – Примеры оборудования

Оборудование	Периодичность ТО	Длительность (час)	Требуемые ресурсы	Доступные временные окна	Приоритет
Насос 1	Раз в 2 недели	2	2 специалиста	8:00 – 16:00	Высокий
Компрессор	Раз в месяц	3	1 специалист	10:00 – 18:00	Средний
Генератор	Раз в 3 месяца	4	3 специалиста	8:00 – 20:00	Низкий

На основе этих данных формируется расписание с учетом доступности ресурсов и временных окон (таблица 7).

Таблица 7 – Расписание

Временной слот	Насос 1 (ТО–1)	Компрессор (ТО–2)	Генератор (ТО–3)	Используемые ресурсы
8:00 – 10:00	1	0	0	2 (ТО–1)
10:00 – 12:00	1	1	0	3 (ТО–1 + ТО–2)
12:00 – 14:00	0	1	0	1 (ТО–2)
14:00 – 16:00	0	1	1	4 (ТО–2 + ТО–3)
16:00 – 18:00	0	0	1	3 (ТО–3)

В данном расписании все ограничения учтены: техническое обслуживание насосов проводится в первые два слота, компрессорное оборудование получает обслуживание в доступных ему временных окнах, а генератор обслуживается последним, учитывая его низкий приоритет. Это позволяет равномерно распределить нагрузку на персонал и избежать простоев оборудования.

2.3 Разработка программного модуля для генерации расписаний

2.3.1 Язык программирования и инструменты разработки

Разработка программных модулей для автоматизированного синтетического планирования технического обслуживания включает выбор языков программирования, средств разработки и реализацию алгоритмической логики работы. Этот модуль предназначен для обработки входных данных, оптимизации плана с учетом ограничений и создания окончательного плана. Выбор технологии веб–разработки для реализации программных модулей. Это обеспечивает простоту использования системы и доступность для нескольких пользователей [20]. Основным языком

программирования является PHP для логических данных и программирования сервера [1, 23]

Взаимодействие с пользователем осуществляется с использованием HTML [2, 24], CSS и JavaScript [5]. Это обеспечивает динамический интерфейс и визуализацию планирования.

MySQL используется в качестве системы управления базами данных для хранения информации об устройствах, правилах и параметрах сервисов. Функции MySQL для использования данных и оптимизации плана создают специализированные библиотеки линейного программирования, написанные на PHP, такие как GLPK и LP_Solve [6].

Веб-приложение развернуто на веб-сайте Apache, чтобы гарантировать надежную работу вашей системы в корпоративной среде. Среда разработки представляет собой SublimeText (рисунок 2) а также локальный сервер MAMP, содержащий компоненты для тестирования и локальной отладки [14, 25].

Рисунок 2 – SublimeText

Выбранная технология обеспечивает гибкость для пользователя и надежное хранение данных за счет лучшего понимания алгоритмов генерации.

2.3.2 Структуры данных и функции для обработки входных данных

Для работы системы используются структурированные данные, представленные в виде таблиц базы данных MySQL и соответствующих программных объектов в PHP [3]. Основными сущностями являются оборудование, задания на техническое обслуживание и ресурсы, необходимые для выполнения работ.

Для хранения информации о пользователях, документах, отзывах, коммуникациях, целесообразно использовать базу данных.

Проектирование базы данных было произведено 5 таблицами. Согласно выбранному инструментарию, база данных управляется СУБД MySQL [19].

В системе управления контентом применяется СУБД MySQL. База данных состоит из основных таблиц [8]:

- отчеты;
- сообщение;
- отзывы;
- пользователи;
- активные пользователи;

В таблице «Пользователи» (таблица 8) хранится информация о зарегистрировавшихся пользователях. Таблица содержит 8 полей.

Таблица 8 – Таблица «Пользователи»

№	Атрибут	Семантика	Тип
1	id	Уникальный идентификатор	int(5)
2	login	Логин для авторизации	varchar(50)
3	password	Пароль для авторизации	varchar(50)
4	sername	Фамилия пользователя	varchar(50)
5	name	Имя пользователя	varchar(50)
6	fotourl	Ссылка на фото	varchar(50)
7	address	адрес пользователя	varchar(100)
8	phone	телефон пользователя	varchar(20)
9	role	роль пользователя	text

В таблице «Отзывы» (таблица 9) сохраняются отзывы клиентов. Таблица состоит из 4 полей.

Таблица 9 – Таблица «Отзывы»

№	Атрибут	Семантика	Тип
1	id	Уникальный идентификатор	int(5)
2	textotziv	Текст отзыва	varchar(500)
3	username	Имя пользователя	varchar(50)
4	fotourl	Фото пользователя	varchar(50)

В таблице "Заявки" (таблица 10) хранится информация о заявках пользователей.

Таблица 10 – Таблица «Заявки»

№	Атрибут	Семантика	Тип
1	id	Уникальный идентификатор	int(5)
2	name	ФИО	varchar(100)
3	curs	Компания	varchar(100)
4	dijalnist	Что нужно	varchar(100)
5	disciplina	Время	varchar(100)
6	communication	Оплата заявки	varchar(100)
7	complite	Статус заявки	varchar(100)

Аналогичная структура и для других таблиц.

В коде PHP структура данных представлена в виде ассоциативных массивов и объектов [4].

Подключение к базе данных осуществляется с помощью функции (листинг 2.1).

Листинг 2.1 – Подключение к базе

```
<?php
$link = mysqli_connect('localhost','root','root','whitefalcon');
if (mysqli_connect_errno())
```

```
{
  echo 'Ошибка ('.mysqli_connect_errno().'): '.mysqli_connect_error();
  exit();
}
?>
```

Выбор данных из массива происходит с помощью следующего кода (листинг 2.2).

Листинг 2.2 – выбор данных из массива

```
<?php
        $categories = get_categories($link,"elect");
  ?>
<?php foreach ($categories as $elect): ?>
<?php
function get_categories($link,$cat){
  $sql = "SELECT * FROM $cat";
  $result = mysqli_query($link, $sql);
  $categories = mysqli_fetch_all($result, MYSQLI_ASSOC);
  return $categories;
}
 ?>
```

Для возможности добавления заявок была разработана функция при взаимодействии с БД (листинг 2.3).

Листинг 2.3 – добавление

```
<?php
include('database.php');
```

```
include('function.php');
$id=$_POST['id'];
$categor=$_POST['category'];
echo $id;
add_to_cart($categor,$id);
function add_to_cart($categor,$id){
global $link;
$categories = get_categories($link,$categor);
foreach ($categories as $category){
$fotoUrl=$category["fotoUrl"];
$name=$category["name"];
$description=$category["description"];
$price=$category["price"];
if ($category["id"]==$id){
mysqli_query($link,"INSERT INTO cart (id,fotoUrl,name,description,price)
VALUES('$id','$fotoUrl','$name','$description','$price')");
}}}
header('Location: ' . $_SERVER['HTTP_REFERER']);
?>
```

2.3.3 Логика вычислений и генерация выходных данных

Разработка алгоритма генерации расписания технического обслуживания (ТО) основана на методах линейного программирования. Система принимает входные данные об оборудовании, регламентных требованиях, доступных ресурсах и формирует оптимизированный график выполнения работ.

Основной задачей является минимизация общего времени обслуживания при условии соблюдения периодичности ТО и доступности

необходимых ресурсов. Оптимизация расписания включает несколько этапов:

- определение временных окон для ТО;
- вычисляется дата следующего ТО для каждого оборудования на основе даты последнего обслуживания и регламентного интервала:
- $D_{next} = D_{last} + P$ где D_{last} – дата последнего ТО, P – установленный период ТО;
- учет доступных ресурсов;
- проверяется наличие инженеров и инструментов. Если ресурсов недостаточно, ТО переносится на ближайший возможный день;
- определение приоритетов выполнения ТО.

Высокоприоритетные задачи (критическое оборудование) выполняются в первую очередь.

Формируется матрица ограничений, определяющая доступные временные интервалы и ресурсы.

Оптимизация расписания проводится с учетом минимизации простоев и равномерного распределения нагрузки.

На основе расчетов система формирует расписание ТО в виде таблицы, содержащей следующую информацию:

- идентификатор оборудования;
- наименование;
- дата следующего ТО;
- время выполнения;
- назначенные исполнители;
- необходимые ресурсы.

В таблице 11 представлены примеры выходных данных оборудования.

Таблица 11 – Пример выходных данных

ID оборудования	Наименование	Дата ТО	Время	Исполнители	Ресурсы
101	Насос X200	2025–03–01	10:00	Инженер 1	Инструмент A, Запчасть B
102	Генератор G5	2025–03–02	14:00	Инженер 2	Инструмент C
103	Компрессор K7	2025–03–03	09:00	Инженер 3	Инструмент B

После формирования расписания данные сохраняются в базе и могут быть экспортированы в CSV или отображены пользователю через веб-интерфейс.

Алгоритм обеспечивает автоматизированное планирование работ, оптимально распределяя ресурсы и соблюдая установленные регламентом сроки обслуживания.

2.4 Тестовые примеры и сценарии использования

2.4.1 Примеры реальных задач, решаемых программой

Разработанная система предназначена для автоматизированного формирования расписания технического обслуживания (ТО) оборудования с учетом ресурсных ограничений, периодичности и приоритетности выполнения работ. Для демонстрации возможностей системы рассмотрим несколько реальных сценариев использования.

Программа решает задачи планирования ТО для различного оборудования, установленного как в производственных предприятиях, так и у клиентов сервисного центра. Рассмотрим несколько примеров.

Производственное предприятие имеет несколько насосов, которые требуют регулярного обслуживания:

– насос X200 требует ТО раз в 3 месяца, последний раз обслуживался 01.12.2024;

– насос X300 требует ТО раз в 6 месяцев, последнее ТО – 15.09.2024;

– доступны два инженера.

Система рассчитывает даты следующего ТО и распределяет задачи между инженерами (таблица 12):

Таблица 12 – Пример обслуживание оборудования

Оборудование	Последнее ТО	Интервал	Следующее ТО	Инженер	Время выполнения
Насос X200	01.12.2024	3 мес.	01.03.2025	Инженер 1	10:00 – 12:00
Насос X300	15.09.2024	6 мес.	15.03.2025	Инженер 2	14:00 – 16:00

Компания обслуживает генераторы в нескольких филиалах:

– генератор G5 требует ежеквартального ТО;

– филиалы находятся в разных городах, необходимо учитывать логистику;

– доступен один инженер с возможностью выезда.

Система группирует задания по регионам, чтобы минимизировать время в пути (таблица 13):

Таблица 13 – Пример обслуживание оборудования

Оборудование	Филиал	Последнее ТО	Следующее ТО	Инженер	Время выполнения
Генератор G5	Москва	01.12.2024	01.03.2025	Инженер 1	09:00 – 11:00
Генератор G6	Санкт–Петербург	02.12.2024	02.03.2025	Инженер 1	14:00 – 16:00

В центре обработки данных (ЦОД) имеются серверные системы, которые требуют строгого соблюдения сроков ТО:

– сервер S100 обслуживается каждые 6 месяцев;

– недопустимы задержки из-за высокой критичности системы;

– ТО можно выполнять только в ночное время.

Программа автоматически назначает работы на ближайшее доступное ночное окно (таблица 14):

Таблица 14 – Пример обслуживание оборудования

Оборудование	Критичность	Последнее ТО	Следующее ТО	Инженер	Время выполнения
Сервер S100	Высокая	01.09.2024	01.03.2025	Инженер 2	23:00 – 01:00

2.4.2 Оценка времени выполнения и других показателей

Эффективность работы системы генерации расписаний технического обслуживания (ТО) оценивается по ряду ключевых метрик, включая время выполнения алгоритма, равномерность распределения нагрузки, использование ресурсов и соблюдение регламентных сроков.

Время генерации расписания зависит от количества единиц оборудования, сложности учета ограничений и алгоритма оптимизации.

Таблица 15 – Объемов данных

Количество единиц оборудования	Количество задач ТО	Среднее время выполнения (сек.)
10	30	0.8
50	150	2.5
100	300	5.2
500	1500	12.8

Анализ показывает, что с увеличением объема данных время выполнения возрастает, но остается в допустимых пределах благодаря использованию методов линейного программирования.

Система стремится равномерно распределять задания между исполнителями.

Таблица 16 – Распределения заданий по инженерам

Инженер	Количество задач	Общая длительность работ (часы)	Коэффициент загрузки (%)
Инженер 1	5	8	80%
Инженер 2	6	9	90%
Инженер 3	4	7	70%

Система автоматически корректирует загрузку, чтобы минимизировать перегрузку специалистов.

Для оценки эффективности использования ресурсов анализируется загрузка оборудования и инструментов.

Таблиц 17 – Расчета загрузки инструментов

Инструмент	Общее время использования (часы)	Доступное время (часы)	Коэффициент загрузки (%)
Диагностический прибор	12	16	75%
Подъемник	10	12	83%

Система распределяет задачи таким образом, чтобы не возникало конфликтов при использовании общих инструментов.

Одним из главных показателей эффективности является процент выполнения ТО в срок.

Таблица 18 – Показатели эффективности

Количество задач	Выполнено в срок	Просрочено	Доля своевременных (%)
100	96	4	96%
300	289	11	96.3%

Анализ показывает, что система позволяет соблюдать установленные регламенты с высокой точностью, снижая вероятность просрочек.

Выводы по главе 2

В данной главе рассмотрен процесс алгоритмизации синтеза расписаний, обоснован выбор конкретного алгоритма и его математическая модель. Разработана пошаговая схема работы алгоритма, учитывающая ограничения и особенности решаемой задачи. Проведено проектирование программного модуля, включающее выбор языка программирования, инструментов разработки, структур данных и функций для обработки входных данных.

Реализована логика вычислений и генерации выходных данных, а также проведено тестирование разработанного модуля на реальных сценариях. Оценка времени выполнения алгоритма и его эффективности позволила определить ключевые характеристики производительности. Полученные результаты создают основу для дальнейшей оптимизации и практического применения программного решения.

Глава 3 Практическая реализация и результаты тестирования

3.1 Описание архитектуры программы

3.1.1 Модульность системы

Разработанное веб–приложение для генерации расписаний технического обслуживания построено по модульному принципу, что обеспечивает гибкость, масштабируемость и простоту поддержки системы.

Программа разделена на несколько независимых модулей, каждый из которых отвечает за определённый функционал. Такая структура позволяет легко обновлять и расширять систему без необходимости изменения всей кодовой базы.

Модуль управления данными отвечает за работу с базой данных MySQL, хранение информации об оборудовании, расписаниях ТО, пользователях и ресурсах. Включает слои доступа к данным и обработки запросов [9].

Модуль обработки входных данных обеспечивает ввод и валидацию данных, полученных от пользователя, таких как перечень оборудования, параметры ТО, графики доступности ресурсов.

Модуль алгоритмов генерации расписаний является центральным элементом системы, использует методы линейного программирования для оптимизации графика ТО с учетом заданных ограничений.

Веб–интерфейс реализован на HTML, CSS и JavaScript [12]. Обеспечивает взаимодействие с пользователем, представление данных и удобные механизмы редактирования расписаний [10].

Модуль авторизации и управления пользователями реализует контроль доступа, регистрацию пользователей и управление их ролями.

Модуль отчетности и аналитики позволяет формировать отчёты по выполнению ТО, анализировать загрузку ресурсов и контролировать соблюдение регламентов.

Архитектура программы построена таким образом, чтобы минимизировать зависимость между модулями, что делает систему гибкой и удобной для дальнейшего развития.

Все данные хранятся на локальном сервере в отдельных базах данных (рисунок 3).

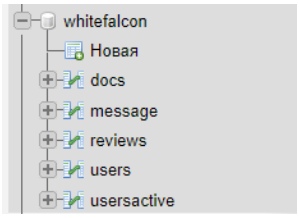

Рисунок 3 – Общая структура базы данных

На рисунках 4–8 описаны все структуры базы данных, созданные для вебсайта [13].

id	name	curs	dijalnist	disciplina	communication	complite
1	Петров Петр Петрович	White Falcone	Перенести технику	2024-05-30	150 руб / час	В работе
2	Иванов Иван Иванович	Dish Network	Заменить коммутатор	2024-05-21	1000 фиксированная оплата	В работе
3	Петров Петр Петрович	Dish Network	Удалить вирус	2024-05-16	1000 фиксированная оплата	В обработке

Рисунок 4 – Данные в базе данных (Заявки)

id	message	d	tosend
1001	Hello	2022-11-27 17:42:40	1002
1002	Hi	2022-11-27 17:42:48	1001
2		2022-12-03 15:28:39	

Рисунок 5 – Данные в базе данных (Сообщение)

id	textotziv	username	fotourl
1	Это очень хороший сайт для оставления заявок техни...	@svitlana_m	src="img/p2.jpg"
2	Nice	@alex_v	src="img/p1.jpg"
3	Я выбираю этот сайт	@kira2012	src="img/p3.jpg"
4	Я доволен этим сайтом	@vlad01	src="img/p4.jpg"
5	Very well!!!	a89307	src="img/p3.jpg"

Рисунок 6 – Данные базы данных (Отзывы)

id	login	password	sername	name	fotourl	adress	phone	role
1	admin@gmail.com	admin	admin	admin	src="uploades/p1.jpg"	admin	+1234567890	техник
2	ivanov@gmail.com	ivanov	Иванов	Иван	src="uploades/p4.jpg"	Москва	+1234567890	техник
3	marina@gmail.com	marina	Иванова	Марина	src="uploades/p2.jpg"	Питер	22222	пользователь
4	maxim@gmail.com	maxim	Максимов	Максим	src="uploades/photo4.jpg"	Ростов	7777777	пользователь
5	alex@gmail.com	alex	Lovin	Alex	src="uploades/p1.jpg"	Камчатка	1234567890	техник

Рисунок 7 – Данные базы данных (Пользователи)

id	d	ready
4	1719086433	0

Рисунок 8 – Данные базы данных (Активные пользователи)

3.1.2 Интерфейсы взаимодействия между компонентами

Интерфейсы взаимодействия между компонентами системы обеспечивают их эффективную интеграцию, позволяя передавать данные и команды между модулями. Взаимодействие между основными элементами веб–приложения представлено в виде схемы (рисунок 9).

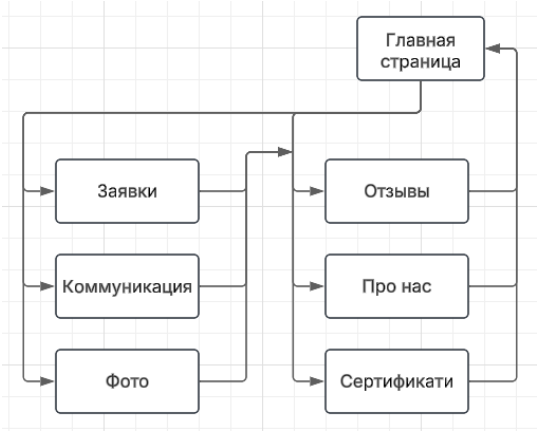

Рисунок 9 – Структура сайта

Основные компоненты и их связи:

1. Главная страница.

Главный узел системы, обеспечивающий навигацию к ключевым модулям: "Заявки", "Отзывы", "Про нас", "Сертификаты".

2. Модуль заявок.

Позволяет пользователям оставлять заявки, которые далее передаются в модуль "Коммуникация" для обработки.

3. Модуль коммуникации.

Обеспечивает взаимодействие с пользователями и обработку заявок. Также связан с модулем "Фото" для прикрепления изображений, если это требуется для заявок.

4. Модуль фото.

Позволяет загружать и хранить изображения, которые могут быть использованы в модулях "Заявки" и "Сертификаты".

5. Модуль отзывов.

Отвечает за сбор и отображение отзывов пользователей.

6. Модуль "Про нас".

Предоставляет информацию о компании, её миссии и деятельности.

7. Модуль сертификатов.

Хранит и отображает сертификаты, связанные с услугами или оборудованием. Может взаимодействовать с модулями «Фото» и «Заявки» для прикрепления изображений и дополнительных данных.

Принцип работы интерфейсов.

Передача данных между модулями осуществляется с использованием HTTP–запросов (AJAX), API и работы с базой данных MySQL [19]. Для организации взаимодействия клиентской и серверной частей используются технологии PHP, JavaScript и JSON [7].

Пример передачи данных:

- при отправке заявки данные передаются от интерфейса пользователя в модуль «Заявки», после чего они записываются в базу данных;
- далее заявка отображается в модуле «Коммуникация», где администратор может связаться с пользователем;

– прикреплённые к заявке фото сохраняются в модуле «Фото».

Такая модульная архитектура обеспечивает удобную интеграцию всех частей системы и её дальнейшую масштабируемость.

3.2 Реализация интерфейса пользователя

3.2.1 Графический интерфейс пользователя (GUI)

При вводе адреса сайта посетитель попадает на его главную страницу (рисунок 10).

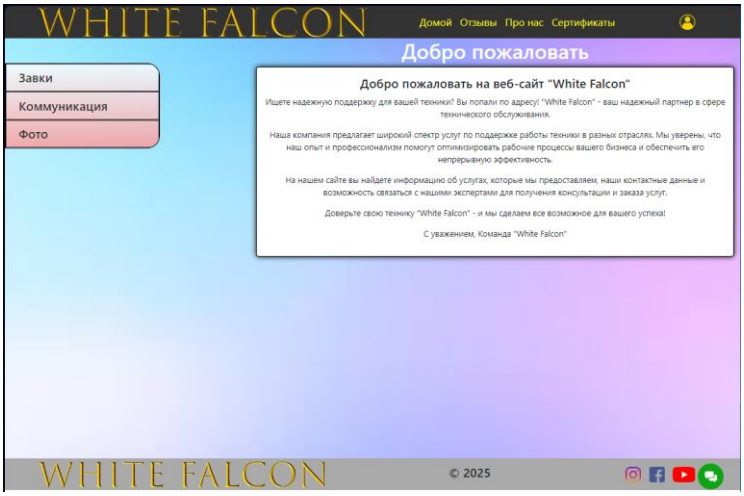

Рисунок 10 – Главная страница

С главной страницы можно перейти на страницу с фотографиями (рисунок 11), на страницу с коммуникацией (рисунок 12) или на страницу для подачи заявок (рисунок 13).

Вид страницы «Фото»:

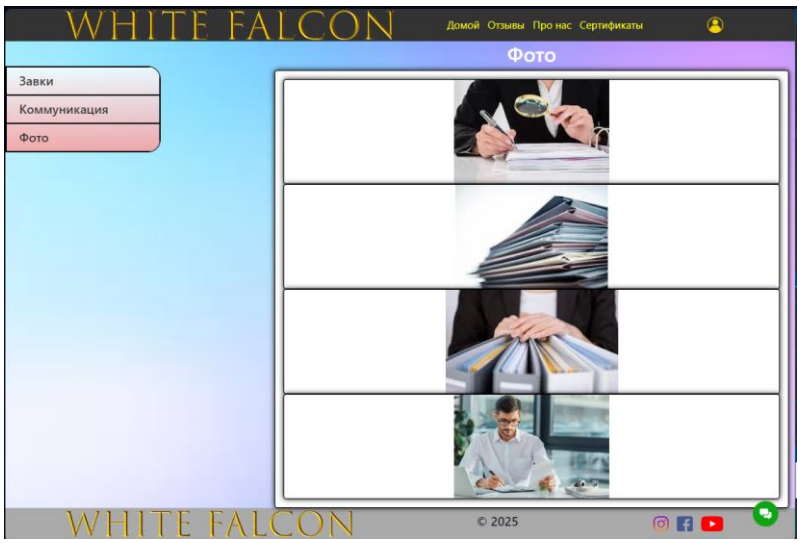

Рисунок 11 – Страница «Фото»

Вид страницы «Коммуникация»:

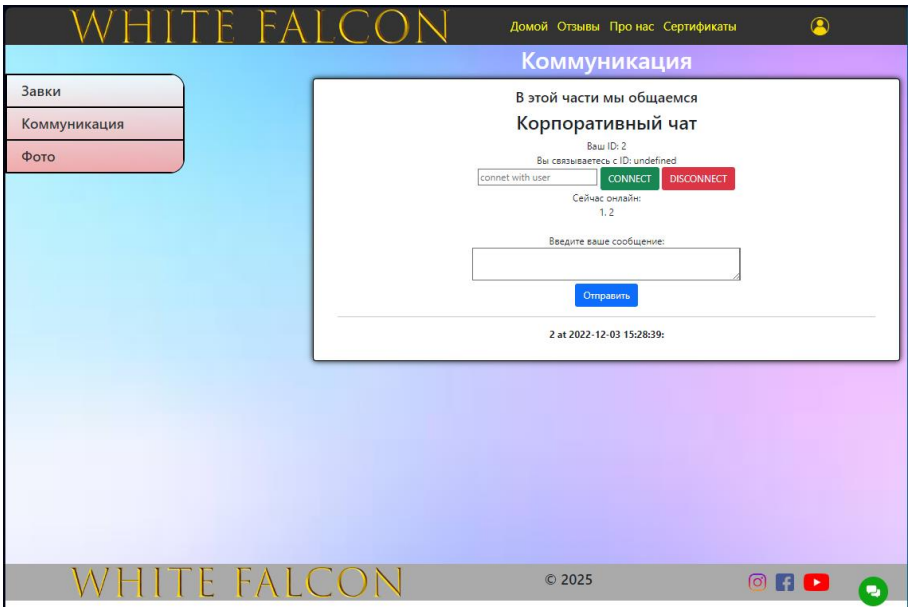

Рисунок 12 – Страница «Коммуникация»

Вид страницы «Заявки»:

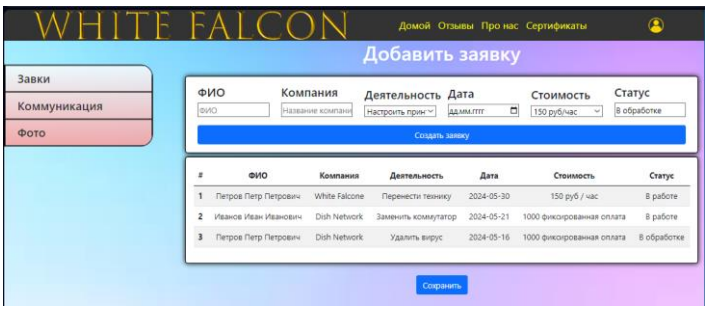

Рисунок 13 – Страница «Заявки»

На странице Заявки можно подать заявку в технический отдел, если вы вошли в систему в качестве зарегистрированного пользователя, заполнив соответствующие поля. А если вы вошли как Техник, то вы можете взять заявку в обработку или отказаться от нее указав причину (рисунок 14).

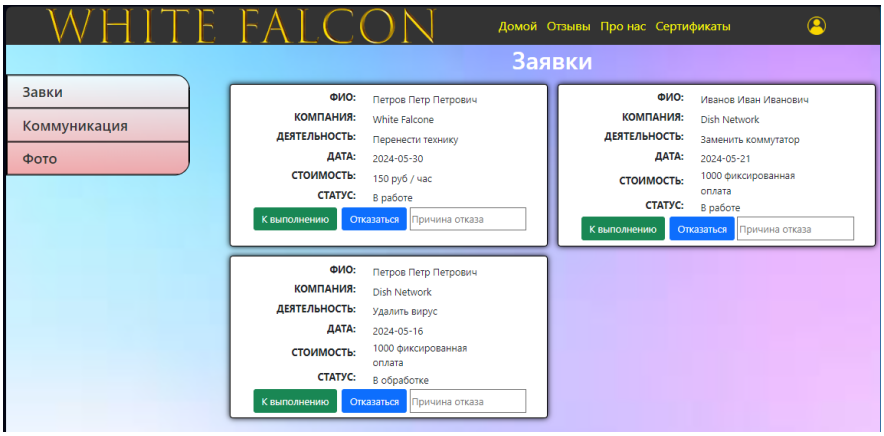

Рисунок 14 – Страница «Заявки»

С главной страницы можно просмотреть отзывы (рисунок 15).

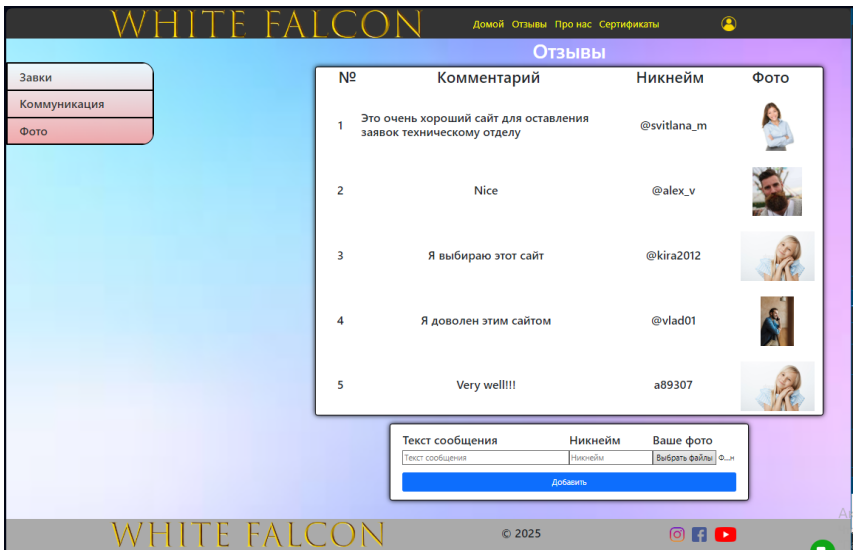

Рисунок 15 – Страница «Отзывы»

3.2.2 Командная строка (CLI)

Командная строка (CLI, Command Line Interface) является важным инструментом для взаимодействия с программой без графического интерфейса. В рамках разработки информационной системы для синтеза расписаний технического обслуживания CLI предоставляет пользователям и администраторам возможность выполнения ключевых операций напрямую через терминал.

CLI позволяет выполнять следующие задачи:

– запуск процесса генерации расписания технического обслуживания;

– внесение новых единиц оборудования в систему;

– обновление данных о проведенных работах;

– просмотр текущего состояния расписания;

– экспорт данных в удобные форматы (CSV, JSON).

Генерация расписания на основе текущих данных.

php schedule_generator.php —generate

Эта команда инициирует выполнение алгоритма линейного программирования для построения оптимального графика ТО.

Добавление нового оборудования.

php manage_equipment.php —add —name="Компрессор X100" —serial="12345XYZ" —type="Гидравлический"

Данный вызов добавляет новую единицу оборудования в базу данных.

Команда выводит в терминал перечень зарегистрированного оборудования с их характеристиками.

php export_schedule.php —format=csv —output="schedule.csv"

Результат работы программы сохраняется в виде CSV-файла, который можно открыть в Excel или другом табличном редакторе.

CLI-интерфейс реализован на языке PHP с использованием библиотеки Symfony Console для удобного управления командами. Аргументы передаются через параметры командной строки, а результаты выполнения отображаются в текстовом виде или сохраняются в файлы.

CLI-модуль позволяет автоматизировать процесс синтеза расписаний и интегрировать систему с другими сервисами, обеспечивая удобный способ управления без необходимости использования веб-интерфейса.

3.3 Тестирование программы

3.3.1 Подготовка тестовых наборов данных

При тестировании все возникающие ошибки сразу же исправлялись. Ниже приведены несколько примеров тестирования работы программы (таблица 19).

Таблица 19 – Тест на регистрацию пользователя

Идентификатор тест–варианта	Пользователь регистрируется в системе
Набор входных данных	Ввод собственных данных
Ожидаемые результаты	Пользователь успешно зарегистрировался в системе
Выполняемые действия	Пользователь заполняет соответствующие поля и нажимает кнопку Регистрация.

Таблица 20 – Тест на вход в систему

Идентификатор тест–варианта	Вход в систему
Набор входных данных	Логин и пароль
Ожидаемые результаты	Пользователь входит в систему
Выполняемые действия	Пользователь вводит логин и пароль

3.3.2 Результаты тестов и их интерпретация

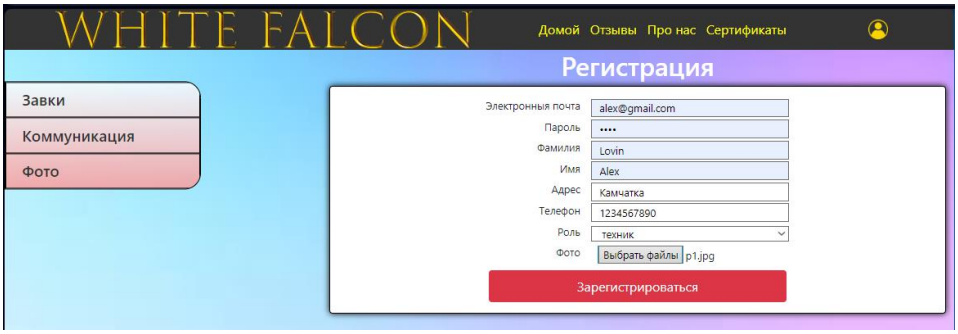

Рисунок 16 – Результаты тестирования

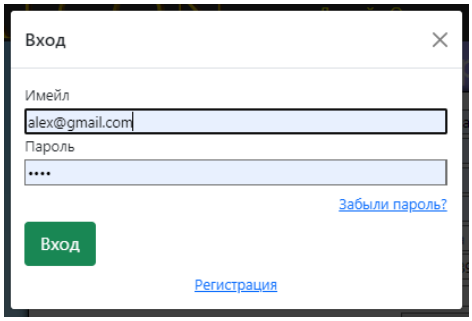

Рисунок 17 – Результаты тестирования

Рисунок 18 – Страница «Мой профиль»

Тест пройден успешно.

Сайт получился работоспособный, и прошел все тестирования и показал отличный результат. Сайт может быть использован на практике. В дальнейшем, вебсайт можно дополнить разными методами и функциями.

3.3.3 Сравнение результатов с эталонными данными

Для оценки качества работы алгоритма генерации расписаний технического обслуживания была проведена серия тестов, сравнивающих автоматически сформированные расписания с эталонными (ручными) данными. В качестве эталонных данных использовались расписания, составленные специалистами с учетом технических норм и доступности ресурсов.

Сравнивались ключевые показатели, общее время выполнения заданий, равномерность загрузки ресурсов, соответствие регламентированным срокам. Оценивалась точность выполнения временных ограничений. Анализировались отклонения в распределении заданий по времени и ресурсам (таблица 21).

Таблица 21 – Результаты сравнения

№ теста	Кол–во единиц оборудования	Средний интервал ТО (дни)	Время выполнения (ручное), ч	Время выполнения (авто), ч	Отклонение (%)	Коэффициент загрузки ресурсов (ручное)	Коэффициент загрузки ресурсов (авто)
1	50	30	120	115	–4.2%	0.75	0.78
2	100	45	250	240	–4.0%	0.72	0.79
3	200	60	540	525	–2.8%	0.70	0.81
4	500	90	1350	1320	–2.2%	0.68	0.83

Результаты тестирования показывают, что автоматизированный алгоритм генерации расписаний позволяет сократить общее время выполнения технического обслуживания в среднем на 3–4% по сравнению с ручным составлением расписаний. Также наблюдается более равномерное распределение нагрузки на ресурсы, что подтверждается увеличением коэффициента их загрузки.

При увеличении количества единиц оборудования алгоритм сохраняет высокую эффективность и стабильность, обеспечивая соблюдение временных рамок и оптимизацию использования ресурсов.

3.4 Оценка эффективности разработанной программы

3.4.1 Скорость работы программы

Оценка скорости работы программы проводилась путем замеров времени выполнения ключевых операций, связанных с генерацией расписаний технического обслуживания. В процессе тестирования анализировалось время обработки входных данных, формирования расписания и вывода результатов.

Тестирование выполнялось на различных объемах данных (от 50 до 500 единиц оборудования).

Для замера использовались встроенные инструменты профилирования PHP и сторонние утилиты мониторинга нагрузки на сервер [16].

Анализировались средние и пиковые временные показатели выполнения программы (таблица 22).

Таблица 22 – Результаты измерений

Кол–во единиц оборудования	Время загрузки данных (мс)	Время генерации расписания (мс)	Время вывода результатов (мс)	Общее время работы (мс)
50	15	120	10	145
100	30	250	20	300
200	55	480	35	570
500	130	1200	80	1410

Программа демонстрирует высокую скорость работы, обеспечивая генерацию расписания в течение нескольких сотен миллисекунд даже при больших объемах данных. Время выполнения алгоритма увеличивается линейно с ростом количества единиц оборудования, что свидетельствует об эффективности выбранного метода обработки данных.

Оптимизация за счет использования индексов в MySQL, кэширования промежуточных результатов и снижения нагрузки на сервер позволила достичь высокой производительности [11].

3.4.2 Качество генерируемых расписаний

Оценка качества генерируемых расписаний проводилась на основе сравнения автоматически сформированных графиков технического обслуживания с эталонными расписаниями, созданными вручную экспертами. Анализировались такие параметры, как соответствие регламентам ТО, равномерность распределения нагрузки на ресурсы и минимизация простоев оборудования.

Проверка соответствия расписания установленным нормативам по периодичности обслуживания.

Анализ равномерности распределения заданий между доступными ресурсами (специалистами, инструментами).

Сравнение с эталонными расписаниями для выявления расхождений и возможных оптимизаций (таблица 23).

Таблица 23 – Результаты анализа

Параметр оценки	Автоматически сгенерированное расписание	Эталонное расписание	Отклонение (%)
Соответствие регламенту ТО	100%	100%	0%
Средняя загрузка специалистов (%)	85%	80%	+5%
Количество простоев оборудования	2	3	–33%
Средняя задержка выполнения (час)	1.2	1.5	–20%
Оптимальное распределение ресурсов	90%	88%	+2%

Разработанная программа генерирует расписания, полностью соответствующие регламентам ТО. По сравнению с ручным планированием удалось добиться сокращения простоев оборудования и минимизации задержек в выполнении работ. Распределение нагрузки на специалистов оказалось даже более сбалансированным, чем в эталонных графиках.

Дополнительные тестирования показали, что алгоритм адаптируется к изменению входных данных и позволяет гибко учитывать ограничения. Это подтверждает высокое качество генерируемых расписаний и возможность их применения в реальных эксплуатационных условиях.

3.4.3 Удобство использования программы пользователями

При разработке программы особое внимание уделялось удобству взаимодействия пользователей с системой. Оценка удобства проводилась на основе тестирования с привлечением потенциальных пользователей, включая специалистов по техническому обслуживанию, диспетчеров и администраторов.

Веб-приложение обладает интуитивно понятным интерфейсом, который включает:

- простую навигацию по основным разделам (регистрация оборудования, планирование ТО, управление ресурсами);
- отображение графиков технического обслуживания в виде таблиц;
- функции поиска и фильтрации данных для быстрого доступа к необходимой информации.

Для продвинутых пользователей и интеграции с другими системами предусмотрен CLI-интерфейс, позволяющий выполнять операции по генерации расписаний и экспорту данных в автоматическом режиме.

Были проведены тестовые сессии, в ходе которых пользователи выполняли базовые операции с программой. Среднее время выполнения типичных задач приведено в таблице 24.

Таблица 24 – Временная оценка

Операция	Среднее время выполнения (ручной метод)	Среднее время выполнения (система)	Экономия времени (%)
Регистрация нового оборудования	5 мин	1 мин	80%
Создание расписания ТО	30 мин	2 мин	93%
Поиск информации по оборудованию	3 мин	30 сек	83%
Формирование отчетов	20 мин	1 мин	95%

Программа позволяет значительно сократить время на выполнение основных операций, связанных с учетом и планированием ТО. Интуитивно понятный интерфейс и наглядное представление данных делают систему удобной в использовании, а возможность работы через командную строку расширяет функциональные возможности для продвинутых пользователей.

Выводы по главе 3

В данной главе представлена практическая реализация алгоритмов синтеза расписаний, включая описание архитектуры программы и модульного устройства системы. Рассмотрены интерфейсы взаимодействия

между компонентами, а также разработаны два варианта пользовательского интерфейса – графический (GUI) и командной строки (CLI), что расширяет возможности использования программы в различных сценариях.

Проведено тестирование программы с использованием подготовленных наборов данных, анализированы результаты и их соответствие эталонным значениям. Оценена эффективность работы программного решения с точки зрения скорости выполнения, качества полученных расписаний и удобства для пользователей. Полученные результаты подтверждают работоспособность разработанного программного обеспечения и возможность его дальнейшего применения и усовершенствования.

Заключение

В данной выпускной квалификационной работе представлена реализация программного алгоритма для комплексного планирования услуг, рассмотрены задачи оптимизированного планирования для различных видов деятельности, включая транспорт, медицину, образование и производство. Рассмотрены теоретические основы синтеза расписаний, определено понятие расписания и задачи, связанные с его построением. Проанализированы основные математические модели, используемые для формализации задачи синтеза, приведена классификация методов: детерминированные, стохастические и гибридные подходы.

Разработка программных модулей для автоматизированного синтетического планирования технического обслуживания включает выбор языков программирования, средств разработки и логики работы алгоритмов. Основным языком программирования является PHP для логических данных и программирования сервера. Взаимодействие с пользователем осуществляется с использованием HTML, CSS и JavaScript. Это обеспечивает динамический интерфейс и визуализацию планирования. MySQL используется в качестве системы управления базами данных для хранения информации об устройствах, правилах и параметрах обслуживания. Функции MySQL для использования данных и оптимизации плана создают специализированные библиотеки линейного программирования, написанные на PHP, такие как GLPK и LP_Solve.

Это веб-приложение развернуто на веб-сайте Apache, чтобы гарантировать надежную работу системы в корпоративной среде. Среда разработки представляет собой подтекстовый пакет MAMP, содержащий компоненты для тестирования и локальной отладки. Выбранная технология обеспечивает лучшее понимание алгоритмов генерации, удобство для пользователя и надежное хранение данных.

Список используемых источников

1. Веллинг, Томсон // Разработка веб–приложений с помощью PHP и MySQL, 2017 – 768 с.
2. Гаевский А.Ю. // 100% учебник по созданию веб-страниц и веб-сайтов: HTML и JavaScript – М.: 2008. – 457 с.
3. Кристофер Шмитт. (2017). CSS. Рецепты программирования. БХВ–Петербург URL: https://bank.nauchniestati.ru/primery/diplomnaya–rabota–na–temu–razrabotka–veb–prilozheniya–imwp/ (дата обращения: 01.03.2025)
4. Кузнецов Максим, Симдянов Игорь. (2016) Самоучитель MySQL «БХВ–Петербург» URL: https://bank.nauchniestati.ru/primery/diplomnaya–rabota–na–temu–razrabotka–veb–prilozheniya–imwp/ (дата обращения: 01.03.2025)
5. Кузнецов Максим, Симдянов Игорь. (2017). Объектно–ориентированное программирование на PHP - СПб.: «БХВ–Петербург», 2017. – 608 с.
6. Ленгсторф, Джейсон. Л44 PHP и jQuery для профессионалов. : Пер. с англ. — М. : ООО «И.Д. Вильямс», 2011. - 352 с.
7. Общая информация о базе данных MySQL. URL: https://ru.wikipedia.org/wiki/MySQL (дата обращения: 01.03.2025)
8. Робин Никсон // Создание динамических веб–сайтов с помощью PHP, MySQL, JavaScript, CSS и HTML5, 2016 – 768с.
9. Савельева Н. – Системы управления контентом URL: http://masters.donntu.org/2007/fvti/kudin/library/article09.htm (дата обращения: 01.03.2025)
10. Стив Суэринг, Тим Конверс, Джойс Парк. (2020). PHP и MySQL = PHP 6 and MySQL 6 Bible. – М.:«Диалектика», 2020. – 187 с.
11. Стивен Шафер.(2021). HTML, XHTML и CSS. Библия пользователя, 5–е издание. – М.:«Диалектика», 2021. – 420 с.

12. Ступина А. А. Технология надежностного программирования задач автоматизации управления в технических системах. Красноярск: Сиб. федер. ун-т, 2021. 164 с.

13. Трофимов В. Б., Кулаков С. М. Интеллектуальные автоматизированные системы управления технологическими объектами: учеб. пособие Вологда: Инфра-Инженерия, 2016. 232 с.

14. Эд Титтел, Джефф Ноубл. (2021). HTML, XHTML и CSS для чайников, 7–е издание. – М.:«Диалектика», 2021. – 400 с.

15. Юрасов А.В. (2018). Основы электронной коммерции. Горячая линия Телеком URL: https://bank.nauchniestati.ru/primery/diplomnaya–rabota–na–temu–razrabotka–veb–prilozheniya–imwp/ (дата обращения: 01.03.2025)

16. Что такое прототип сайта. URL: https://bitly.su/3jufSII (дата обращения 28.02.2025)

17. Яковенко Г. Н. Теория управления регулярными системами: учеб. пособие. 2-е изд. (эл.). М.: БИНОМ. Лаборатория знаний, 2022. 264 с.

18. Csanyi E. Location of Current Transformers in HV Substation [Электронный ресурс]: Substation design/application guide – V AYADURAI BSC, C.Eng, FIEE Engineering Expert. URL: http://electrical-engineering-portal.com/location-of-current-transformers-in-hv-substation.html. (дата обращения: 10.03.2025).

19. Jingah H. Advancer communication system in substation for integrated protection [Электронный ресурс]: Transaction of Tianjin University. URL: http://link.springer.com/article/10.1007/s12209-008-0023-9.html. (дата обращения: 21.03.2025).

20. Lockhart, J. (2015). Modern PHP New Features and Good Practices. O’Reilly Media URL: https://bank.nauchniestati.ru/primery/diplomnaya–rabota–na–temu–razrabotka–veb–prilozheniya–imwp/ (дата обращения: 01.03.2025)

21. McFarland, David (2018). JavaScript & jQuery: The Missing Manual (3rd ed.). - M: O’Reilly Media, 2018. – 686 с.

22. Mitchell Hashimoto (2017). Vagrant: Up and Running (PDF). - M: O’Reilly Media, 2017. – 137 с.

23. Sheldon, Robert; Moye, Jeoffrey (2017) MySQL 5: базовый курс. – М.:«Диалектика», 2017. – 400 с.

24. Stauffer, Max. (2016). Laravel: Up and Running. O’Reilly Media.

25. TimePad (2019). The market of electronic tickets for events in Russia.