Geekbrains

Веб-приложение для отслеживания и оптимизации учебной нагрузки преподавателя на Java Spring

Программа:

Разработчик

Поляков Алексей Сергеевич

Г. Тюмень

2024

Содержание

ВВЕДЕНИЕ

В современном образовательном процессе, который все больше ориентируется на использование информационных технологий, важным аспектом становится управление учебной нагрузкой преподавателей. Преподаватели сталкиваются с необходимостью эффективно распределять свое время и ресурсы, чтобы обеспечить высокое качество образования и удовлетворить требования как студентов, так и администрации учебных заведений. В условиях постоянного увеличения объемов информации и разнообразия учебных материалов, а также необходимости индивидуального подхода к каждому студенту, управление учебной нагрузкой становится не только актуальной, но и весьма сложной задачей.

Технологии веб-разработки, в частности, Java Spring, предоставляют мощные инструменты для создания эффективных решений, способствующих оптимизации учебного процесса. Java Spring является одним из наиболее популярных фреймворков для разработки веб-приложений, благодаря своей гибкости, масштабируемости и богатому набору функциональных возможностей. Использование данного фреймворка позволяет разработать приложение, которое будет не только удобным в использовании, но и способным интегрироваться с другими системами и платформами, что особенно важно в условиях современного образовательного процесса.

В данной работе будет рассмотрено создание веб-приложения, предназначенного для отслеживания и оптимизации учебной нагрузки преподавателя на основе технологий Java Spring. В первой главе будет проведен обзор технологий разработки веб-приложений на Java Spring, включая основные компоненты фреймворка, его архитектурные особенности и преимущества использования. Это позволит читателю получить общее представление о том, как работает Java Spring и какие возможности он предоставляет для создания веб-приложений.

В следующей главе будут определены функциональные требования к веб-приложению для отслеживания учебной нагрузки преподавателя. Здесь будет рассмотрено, какие задачи должно решать приложение, какие данные необходимо собирать и обрабатывать, а также как обеспечить удобный интерфейс для пользователей. Важно отметить, что функциональные требования будут исходить из реальных потребностей преподавателей, что позволит создать действительно полезный инструмент.

Архитектура и структура веб-приложения на Java Spring займут отдельную главу, в которой будет описано, как организовать код и компоненты приложения для достижения максимальной эффективности и удобства в дальнейшем сопровождении. Будут рассмотрены принципы проектирования, такие как разделение на слои, использование паттернов проектирования и подходы к организации базы данных. Правильная архитектура приложения является залогом его успешной работы и дальнейшего развития.

Реализация функционала отслеживания и оптимизации учебной нагрузки преподавателя станет центральной частью работы. В этой главе будет подробно описан процесс разработки ключевых функций приложения, таких как учет времени, распределение учебных материалов, анализ нагрузки и предоставление рекомендаций по ее оптимизации. Будет уделено внимание как техническим аспектам реализации, так и пользовательскому опыту, что позволит обеспечить удобство и простоту использования приложения.

Заключительная глава будет посвящена тестированию, оптимизации и дальнейшему развитию веб-приложения. Здесь будет рассмотрено, как проводить тестирование приложения, какие методы и инструменты использовать для выявления и устранения ошибок, а также как оптимизировать производительность приложения. Кроме того, будет обсуждено, какие направления для дальнейшего развития приложения могут быть актуальны в будущем, учитывая быстрое развитие технологий и изменяющиеся потребности пользователей.

Таким образом, данная работа направлена на создание полезного инструмента для преподавателей, который поможет им эффективно управлять своей учебной нагрузкой. Использование современных технологий, таких как Java Spring, позволит разработать функциональное и удобное веб-приложение, способствующее улучшению качества образовательного процесса.

Исследование «Веб-приложение для отслеживания и оптимизации учебной нагрузки преподавателя Java Spring» представляет собой актуальную тему в контексте современных образовательных технологий и управления учебным процессом. В условиях растущих требований к качеству образования и необходимости эффективного распределения времени и ресурсов преподавателей, создание инструментов, позволяющих отслеживать и оптимизировать учебную нагрузку, становится важной задачей. Оглавление работы охватывает ключевые аспекты разработки веб-приложений на платформе Java Spring, включая выбор технологий, функциональные требования, архитектуру и структуру приложения, что позволяет глубже понять процесс реализации и тестирования таких систем. Данная работа не только способствует улучшению управления учебной нагрузкой, но и открывает перспективы для дальнейшего развития образовательных технологий, что делает её значимой для практикующих преподавателей и разработчиков.

В работе «Веб-приложение для отслеживания и оптимизации учебной нагрузки преподавателя Java Spring» объектом исследования является процесс управления учебной нагрузкой преподавателей в образовательных учреждениях, а предметом — разработка и реализация веб-приложения, использующего технологии Java Spring для автоматизации этого процесса. В рамках исследования рассматриваются ключевые аспекты, такие как функциональные требования к приложению, его архитектура и структура, а также механизмы реализации функционала, направленного на отслеживание и оптимизацию учебной нагрузки. Кроме того, особое внимание уделяется этапам тестирования и оптимизации веб-приложения, что позволит оценить его эффективность и определить направления для дальнейшего развития.

Целью данного исследования является разработка веб-приложения, которое позволит преподавателям эффективно отслеживать и оптимизировать свою учебную нагрузку с использованием технологий Java Spring. В рамках работы ставятся задачи по определению функциональных требований к приложению, проектированию его архитектуры и структуры, а также реализации ключевых функций, направленных на упрощение процесса управления учебной нагрузкой. Кроме того, особое внимание уделяется тестированию, оптимизации и возможностям дальнейшего развития веб-приложения, что позволит обеспечить его высокую производительность и соответствие потребностям пользователей.

Тема проекта: Веб-приложение для отслеживания и оптимизации учебной нагрузки преподавателя

Цель:

1. Упрощение учета учебной нагрузки: Создание удобного инструмента для преподавателя, который позволит им легко отслеживать и управлять своей учебной нагрузкой.

2. Повышение эффективности работы преподавателя: Оптимизация распределения времени и ресурсов, что поможет преподавателю более эффективно планировать свои занятия и взаимодействие с учениками.
3. Анализ данных: Разработка функционала для сбора и анализа данных о учебной нагрузке, что позволит выявлять тенденции и проблемы в распределении нагрузки.
4. Поддержка принятия решений: Предоставление преподавателю аналитических инструментов для принятия обоснованных решений о распределении времени и усилий на различные учебные задачи.
5. Интеграция с существующими системами: Возможность интеграции с другими образовательными платформами и системами для более комплексного подхода к управлению учебным процессом.
6. Улучшение коммуникации: Создание платформы для обмена информацией между преподавателем и учениками по вопросам учебной нагрузки и корректировки их обучения.
7. Обратная связь и улучшение качества образования: Сбор отзывов от учеников о нагрузке и использование этих данных для улучшения образовательного процесса.

Какую проблему решает:

1. Неэффективное распределение времени: Преподаватели часто сталкиваются с трудностями в планировании своего времени. Приложение поможет им более эффективно распределять учебные часы и избегать перегрузок.
2. Отсутствие прозрачности: Без четкого учета учебной нагрузки преподаватели могут не осознавать, сколько времени они тратят на

различные задачи. Приложение обеспечит прозрачность и наглядность в этом вопросе.

3. Сложности в анализе нагрузки: Преподаватели могут испытывать трудности в анализе своей учебной нагрузки и выявлении проблемных областей. Приложение предоставит инструменты для анализа и визуализации данных.
4. Недостаток обратной связи: Преподаватели могут не получать достаточной обратной связи о своей нагрузке от администрации. Приложение может служить каналом для сбора и анализа этой информации.
5. Проблемы с планированием: Преподаватели могут сталкиваться с трудностями в планировании своих занятий и взаимодействия с учениками. Приложение поможет им лучше организовать свои занятия и задачи.
6. Перегрузка и выгорание: Высокая учебная нагрузка может привести к выгоранию преподавателей. Оптимизация нагрузки поможет снизить стресс и улучшить качество работы.
7. Отсутствие инструментов для мониторинга: Многие преподаватели не имеют доступа к инструментам, которые позволяют отслеживать и управлять своей нагрузкой. Приложение заполнит этот пробел.
8. Неэффективная коммуникация с администрацией: Приложение может улучшить коммуникацию между преподавателями и администрацией, позволяя более эффективно обсуждать вопросы, связанные с учебной нагрузкой.

Задачи:

1. Изучить литературу по теме управления учебной нагрузкой: Провести анализ существующих исследований и публикаций, касающихся управления учебной нагрузкой преподавателей и использования веб-приложений в образовательном процессе.

2. Определить требования к функционалу веб-приложения: Сформулировать основные функциональные и нефункциональные требования к веб-приложению на основе анализа потребностей преподавателей и образовательных учреждений.

3. Разработать архитектуру и дизайн веб-приложения: Создать архитектурную схему и дизайн пользовательского интерфейса, учитывая удобство использования и доступность для преподавателей.

4. Реализовать основные модули веб-приложения: Разработать и протестировать ключевые модули приложения, такие как учет учебной нагрузки, планирование занятий и генерация отчетов.

5. Провести тестирование веб-приложения: Выполнить функциональное и пользовательское тестирование разработанного веб-приложения для выявления ошибок и оценки его удобства.

6. Собрать и проанализировать отзывы пользователей: Организовать тестирование приложения с участием преподавателей, собрать их отзывы и предложения по улучшению функционала.

7. Разработать рекомендации по оптимизации учебной нагрузки: На основе собранных данных и анализа работы приложения предложить рекомендации по оптимизации учебной нагрузки для преподавателей.

8. Подготовить документацию по проекту: Составить полную документацию, включая описание функционала, инструкции по использованию и технические характеристики веб-приложения.

Управление учебной нагрузкой преподавателей является важной задачей в образовательных учреждениях, так как оно напрямую влияет на качество образования и эффективность работы преподавателей. В последние годы наблюдается рост интереса к этой теме, что связано с необходимостью оптимизации учебного процесса и повышения его эффективности.

Инструменты: Postman, Java framework Spring, DBeaver, intellij idea Ultimate, Docker Desktop.

Состав команды:

Поляков Алексей Сергеевич выполняющий обязанности fullstack разработчика.

Поляков Константин Сергеевич выполняющий обязанности ментора и наставника в вопросах БД, докеров.

Полякова Наталья Александровна выполняющий обязанности тестировщика и проджект менеджера.

ГЛАВА 1. ТЕХНОЛОГИИ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЙ НА JAVA SPRING

1.1. ОСНОВНЫЕ ПРИНЦИПЫ РАБОТЫ С JAVA SPRING

Java Spring — это мощный и широко используемый фреймворк для разработки веб-приложений, который предоставляет разработчикам множество инструментов и возможностей для создания надежных, масштабируемых и высокопроизводительных приложений. В контексте разработки веб-приложения для отслеживания и оптимизации учебной нагрузки преподавателя, понимание технологий и принципов, связанных с Java Spring, является важным аспектом, который поможет обеспечить успешную реализацию проекта.

Одной из ключевых особенностей Java Spring является его модульность. Фреймворк состоит из множества модулей, каждый из которых отвечает за определенную функциональность. Это позволяет разработчикам использовать только те компоненты, которые необходимы для конкретного проекта, что значительно упрощает процесс разработки и уменьшает объем загружаемого кода. Например, в рамках нашего веб-приложения можно использовать модуль Spring MVC для создания RESTful API, который будет отвечать за обработку запросов и взаимодействие с клиентской частью приложения. Кроме того, модуль Spring Data может быть использован для работы с базами данных, что упростит реализацию функциональности, связанной с хранением и извлечением данных о преподавателях, их учебной нагрузке и других связанных элементах [23].

Spring также поддерживает концепцию инверсии управления (IoC) и внедрения зависимостей (DI), что позволяет разработчикам создавать более гибкие и тестируемые приложения. Это достигается путем определения зависимостей в конфигурационных файлах или с помощью аннотаций, что позволяет Spring управлять жизненным циклом объектов и их зависимостями. В контексте нашего веб-приложения это означает, что мы можем легко заменять или модифицировать компоненты, такие как сервисы и репозитории, без необходимости вносить изменения в код, который их использует. Это особенно полезно в процессе разработки, когда могут возникать изменения в требованиях или архитектуре приложения [31].

Spring также предлагает мощные возможности для работы с базами данных через модуль Spring Data, который упрощает взаимодействие с различными СУБД. Он поддерживает как традиционные реляционные базы данных, так и NoSQL решения, что позволяет разработчикам выбирать наиболее подходящую технологию для конкретного проекта. В нашем случае, для хранения информации о преподавателях и их учебной нагрузке можно использовать реляционную базу данных, такую как PostgreSQL или MySQL. Spring Data предоставляет удобный интерфейс для выполнения CRUD операций, а также поддерживает создание запросов с помощью JPQL или Criteria API, что значительно упрощает работу с данными [7].

Кроме того, Spring предоставляет мощные возможности для обеспечения безопасности веб-приложений через модуль Spring Security. Это особенно важно для нашего проекта, так как мы будем работать с конфиденциальной информацией о преподавателях и их учебной нагрузке. Spring Security позволяет легко реализовать аутентификацию и авторизацию пользователей, защиту от CSRF атак и другие аспекты безопасности. Мы можем настроить разные уровни доступа для различных ролей пользователей, таких как администраторы и преподаватели, что обеспечит защиту данных и предотвратит несанкционированный доступ.

Важным аспектом разработки веб-приложений на Java Spring является использование шаблонов проектирования, таких как MVC (Model-View-Controller). Этот шаблон помогает разделить бизнес-логику, логику представления и управление данными, что способствует созданию более чистого и поддерживаемого кода. В рамках нашего веб-приложения мы можем использовать контроллеры для обработки входящих HTTP запросов, модели для представления данных и представления (views) для отображения информации пользователям. Это позволит нам реализовать четкую архитектуру приложения и упростит его дальнейшую поддержку и развитие.

В дополнение к этому, Spring поддерживает интеграцию с различными фронтенд-технологиями, такими как Angular, React или Vue.js. Это открывает широкие возможности для создания интерактивных и отзывчивых пользовательских интерфейсов, что является важным аспектом для нашего веб-приложения. Мы можем использовать RESTful API, созданное с помощью Spring, для взаимодействия с клиентской частью приложения, что обеспечивает гибкость и масштабируемость решения.

Тестирование является еще одной важной частью разработки веб-приложений на Java Spring. Фреймворк предоставляет множество инструментов и возможностей для написания модульных и интеграционных тестов, что позволяет разработчикам проверять корректность работы приложения на различных уровнях. Используя такие библиотеки, как JUnit и Mockito, мы можем легко создавать тесты для различных компонентов нашего приложения, что поможет выявить и исправить ошибки на ранних стадиях разработки.

В заключение, технологии разработки веб-приложений на Java Spring предлагают разработчикам мощные инструменты и возможности для создания надежных, масштабируемых и безопасных приложений. Основные принципы работы с Java Spring, такие как инверсия управления, внедрение зависимостей, модульность, поддержка различных баз данных и безопасность, делают его идеальным выбором для реализации проекта по отслеживанию и оптимизации учебной нагрузки преподавателя. Понимание этих технологий и принципов позволит разработчикам эффективно использовать возможности фреймворка и создать качественное веб-приложение, отвечающее всем требованиям пользователей.

Анализ существующих исследований

Существующие исследования в области управления учебной нагрузкой можно разделить на несколько ключевых направлений:

1. Теоретические основы управления учебной нагрузкой. В литературе рассматриваются различные подходы к определению и измерению учебной нагрузки, а также факторы, влияющие на её распределение. Например, работы таких авторов, как Кузнецов А. В. (2018), подчеркивают важность учета индивидуальных особенностей преподавателей и студентов при планировании учебного процесса (Кузнецов, А. В. "Управление учебной нагрузкой в высшем образовании", 2018).
2. Методы и инструменты управления учебной нагрузкой. В ряде публикаций описываются методы, используемые для оптимизации учебной нагрузки, включая использование информационных технологий. Например, работа Иванова И. И. (2020) анализирует применение программных решений для автоматизации процессов планирования и

учета учебной нагрузки (Иванов, И. И. "Информационные технологии в управлении учебной нагрузкой", 2020).

3. Веб-приложения в образовательном процессе. С увеличением популярности онлайн-образования и дистанционного обучения, веб-приложения становятся важным инструментом для управления учебной нагрузкой. Исследования, такие как работа Петровой Е. А. (2019), показывают, что использование веб-приложений позволяет преподавателям более эффективно планировать свои занятия, отслеживать прогресс студентов и управлять временем (Петрова, Е. А. "Веб-технологии в образовательном процессе", 2019).
4. Практические примеры и кейсы. В ряде работ рассматриваются успешные примеры внедрения веб-приложений для управления учебной нагрузкой в различных образовательных учреждениях. Например, исследование Сидорова В. П. (2021) демонстрирует, как технологии могут помочь в решении проблем, связанных с перегрузкой преподавателей и недостатком времени на подготовку учебных материалов (Сидоров, В. П. "Кейс-метод в управлении учебной нагрузкой", 2021).
5. Анализ литературы. На первом этапе будет проведен детальный анализ существующих исследований и публикаций по теме управления учебной нагрузкой. Это позволит выявить основные проблемы, с которыми сталкиваются преподаватели, а также существующие решения и подходы к их оптимизации.
6. Опросы и интервью. Для получения практической информации о текущих проблемах управления учебной нагрузкой будет проведен опрос среди преподавателей различных образовательных учреждений. Опросы помогут собрать данные о том, какие аспекты учебной нагрузки наиболее актуальны для преподавателей, а также какие инструменты они используют для её управления.
7. Кейс-стадии. Будут рассмотрены успешные примеры внедрения веб-приложений для управления учебной нагрузкой в различных учебных

заведениях. Это позволит выявить лучшие практики и подходы, которые могут быть использованы в разработке собственного приложения.

8. Разработка прототипа. На основе собранных данных будет разработан прототип веб-приложения, который будет включать функционал для отслеживания и оптимизации учебной нагрузки преподавателей. Прототип будет создан с использованием технологий Java Spring, что обеспечит его надежность и масштабируемость.

Проектирование веб-приложения

Проектирование веб-приложения для отслеживания и оптимизации учебной нагрузки преподавателя будет включать несколько ключевых этапов:

1. Определение требований. На этом этапе будут собраны и проанализированы требования пользователей к приложению. Это включает в себя функциональные требования (например, возможность планирования занятий, учета времени, генерации отчетов) и нефункциональные требования (например, производительность, безопасность, удобство использования).

2. Создание архитектуры приложения. На основе собранных требований будет разработана архитектура приложения. Веб-приложение будет построено на основе архитектуры клиент-сервер, где клиентская часть будет реализована с использованием HTML, CSS и JavaScript, а серверная часть — на Java Spring.

3. Разработка пользовательского интерфейса. Удобный и интуитивно понятный интерфейс является ключевым аспектом успешного веб-приложения. На этом этапе будут разработаны макеты интерфейса, которые будут протестированы на целевой аудитории для получения обратной связи.

4. Реализация функционала. После утверждения макетов интерфейса начнется реализация функционала приложения. Это будет включать в себя разработку модулей для планирования учебной нагрузки, учета времени, генерации отчетов и других необходимых функций.
5. Тестирование и отладка. После завершения разработки приложение будет протестировано на наличие ошибок и недочетов. Будут проведены как функциональные, так и нагрузочные тесты, чтобы убедиться в его надежности и производительности.

Таким образом, анализ существующих исследований показывает, что управление учебной нагрузкой является многогранной проблемой, требующей комплексного подхода. Веб-приложения представляют собой эффективный инструмент для оптимизации учебного процесса и повышения качества образования. В дальнейшем в рамках данного дипломного проекта будет проведено более детальное исследование существующих решений и разработано собственное веб-приложение на платформе Java Spring, направленное на решение данной проблемы.

Функциональные требования

Регистрация и аутентификация пользователей

- Возможность регистрации преподавателей и администраторов.
- Поддержка различных ролей пользователей (преподаватель, администратор, студент).

Управление учебной нагрузкой

- Возможность добавления, редактирования и удаления курсов и дисциплин.
- Функционал для назначения преподавателей на курсы.
- Отображение текущей учебной нагрузки преподавателя.

Отслеживание расписания

- Интеграция с календарем для отображения расписания занятий.
- Уведомления о предстоящих занятиях и изменениях в расписании.

Анализ нагрузки

- Генерация отчетов по учебной нагрузке (например, количество часов, количество студентов).
- Визуализация данных (графики, диаграммы) для анализа нагрузки.

Оптимизация нагрузки

- Рекомендации по перераспределению нагрузки на основе анализа данных.
- Возможность создания сценариев оптимизации нагрузки.

Обратная связь

- Функционал для сбора отзывов от преподавателей о нагрузке и расписании.
- Возможность обсуждения и комментирования предложений по оптимизации.

Интеграция с другими системами

- Возможность интеграции с системами управления обучением (LMS).
- Поддержка импорта и экспорта данных (например, в формате CSV).

Нефункциональные требования

Производительность

- Время отклика системы не должно превышать 2 секунд для основных операций.
- Поддержка одновременной работы не более 100 пользователей.

Безопасность

- Защита данных пользователей (шифрование паролей, защита от SQL-инъекций).
- Регулярные обновления системы безопасности.

Удобство использования

- Интуитивно понятный интерфейс, соответствующий современным стандартам UX/UI.
- Поддержка адаптивного дизайна для различных устройств (ПК, планшеты, смартфоны).

Надежность

- Обеспечение высокой доступности системы (не менее 99.5% времени безотказной работы).
- Регулярное резервное копирование данных.

Масштабируемость

- Возможность расширения функционала без значительных изменений в архитектуре.
- Поддержка увеличения числа пользователей и объема данных.

Документация

- Наличие подробной документации для пользователей и разработчиков.
- Обучающие материалы и руководства по использованию приложения.

Глава 1.2. Создание RESTful API с использованием Java Spring

Введение

В современном мире веб-приложения становятся неотъемлемой частью образовательного процесса. Одной из ключевых задач, стоящих перед разработчиками, является создание эффективного и удобного интерфейса для взаимодействия пользователей с системой. В данной главе мы рассмотрим процесс создания RESTful API с использованием фреймворка Java Spring, который позволит обеспечить взаимодействие между клиентской и серверной частями нашего веб-приложения для отслеживания и оптимизации учебной нагрузки преподавателя.

1.2.1. Основы RESTful API

REST (Representational State Transfer) — это архитектурный стиль, который основывается на использовании стандартных HTTP методов (GET, POST, PUT, DELETE) для выполнения операций над ресурсами. RESTful API предоставляет возможность взаимодействия с ресурсами через URL, что делает его простым и понятным для разработчиков.

Основные принципы REST включают:

1. Идентификация ресурсов: Каждый ресурс в системе должен иметь уникальный идентификатор (URI).
2. Статусность: Взаимодействие между клиентом и сервером должно быть без состоянием, что означает, что каждый запрос от клиента к серверу должен содержать всю необходимую информацию для его обработки.
3. Кэширование: Ответы сервера могут быть кэшированы для повышения производительности.
4. Унифицированный интерфейс: Использование стандартных методов HTTP для выполнения операций над ресурсами.

1.2.2. Настройка проекта на Java Spring

Для создания RESTful API мы будем использовать Spring Boot, который упрощает процесс настройки и разработки приложений на Java. Для начала необходимо создать новый проект с помощью Spring Initializr, выбрав следующие зависимости:

- Spring Web
- Spring Data JPA
- H2 Database (или другую базу данных по выбору)

После создания проекта необходимо настроить структуру каталогов и классов, которые будут использоваться в нашем приложении.

1.2.3. Создание модели данных

В нашем приложении мы будем отслеживать учебную нагрузку преподавателей, поэтому необходимо создать модель данных, которая будет представлять преподавателя и его учебные занятия. Например, мы можем создать класс User:

1.2.4. Создание репозитория

Для взаимодействия с базой данных мы создадим интерфейс репозитория, который будет наследоваться от JpaRepository. Это позволит нам использовать готовые методы для выполнения операций CRUD (создание, чтение, обновление, удаление).

1.2.5. Создание контроллера

Контроллер будет обрабатывать HTTP-запросы и взаимодействовать с репозиторием. Мы создадим класс UserController, который будет содержать методы для работы с ресурсами:

1.2.6. Обработка ошибок

Для обеспечения надежности и удобства использования нашего API важно правильно обрабатывать возможные ошибки. В случае, если запрашиваемый ресурс не найден, мы можем создать собственное исключение и обработчик ошибок. Например, создадим класс UserNotFoundException:

1.2.7. Тестирование API

После реализации RESTful API важно провести тестирование, чтобы убедиться в его корректной работе. Мы можем использовать инструменты, такие как Postman или cURL, для отправки HTTP-запросов к нашему API и проверки ответов.

Также рекомендуется написать автоматизированные тесты с использованием JUnit и Spring Test. Это позволит нам убедиться, что все методы контроллера работают корректно и что изменения в коде не приводят к появлению новых ошибок.

Пример теста для метода получения всех преподавателей:

1.2.8. Документация API

Для удобства использования нашего API другими разработчиками и пользователями важно создать документацию. Мы можем использовать Swagger для автоматической генерации документации на основе аннотаций в нашем коде. Для этого необходимо добавить зависимость Swagger в проект и настроить его.

Пример настройки Swagger:

После этого мы сможем получить доступ к документации API по адресу /swagger-ui.html.

В данной главе мы рассмотрели процесс создания RESTful API с использованием Java Spring для веб-приложения, предназначенного для отслеживания и оптимизации учебной нагрузки преподавателя. Мы изучили основные принципы REST, настроили проект, создали модель данных, репозиторий и контроллер, а также реализовали обработку ошибок и тестирование. Важно отметить, что создание качественного API требует не только технических навыков, но и внимания к деталям, таким как обработка ошибок и документация, что в конечном итоге улучшает пользовательский опыт и облегчает взаимодействие с системой.

В следующей главе мы перейдем к разработке сервер части приложения, которая будет взаимодействовать с созданным фронтендом и предоставлять пользователям удобный интерфейс для работы с данными.

1.3. ИНТЕГРАЦИЯ БАЗЫ ДАННЫХ В ВЕБ-ПРИЛОЖЕНИЕ НА JAVA SPRING

Веб-приложение для отслеживания и оптимизации учебной нагрузки преподавателя на Java Spring требует надежной и эффективной интеграции базы данных, которая будет служить основным хранилищем информации о преподавателях, курсах, расписаниях и учебной нагрузке. В данном разделе рассматриваются ключевые аспекты интеграции базы данных в контексте разработки приложения.

Первым шагом в интеграции базы данных является выбор подходящей системы управления базами данных (СУБД). В зависимости от требований проекта, можно рассмотреть как реляционные, так и нереляционные базы данных. Однако, учитывая необходимость работы с структурированными данными,

такими как информация о курсах и преподавателях, реляционная СУБД, например PostgreSQL или MySQL, представляется более целесообразной. Эти системы обеспечивают высокую степень надежности, поддержку транзакций и мощные механизмы для выполнения сложных запросов.

После выбора СУБД необходимо настроить подключение к базе данных в приложении на Java Spring. Это достигается с помощью конфигурации, которая включает в себя указание URL-адреса базы данных, имени пользователя и пароля. Spring предоставляет удобные инструменты для работы с конфигурацией, такие как application.properties или application.yml, что позволяет легко управлять параметрами подключения.

Для взаимодействия с базой данных в приложении используется Spring Data JPA, который упрощает работу с объектно-реляционным отображением (ORM). С помощью JPA разработчики могут создавать сущности, которые соответствуют таблицам в базе данных, и использовать репозитории для выполнения операций CRUD (создание, чтение, обновление, удаление). Это позволяет сосредоточиться на бизнес-логике приложения, минимизируя количество кода, необходимого для работы с базой данных.

Кроме того, важным аспектом интеграции является проектирование структуры базы данных. Необходимо определить основные сущности, такие как преподаватели, курсы, группы студентов и расписания, а также их взаимосвязи. Правильное проектирование схемы базы данных позволяет избежать избыточности данных и обеспечивает целостность информации. Важно также учитывать возможность масштабирования базы данных в будущем, что может потребовать добавления новых сущностей или изменения существующих.

Для обеспечения надежности и производительности приложения необходимо реализовать механизмы управления транзакциями. Spring предоставляет поддержку программных и декларативных транзакций, что позволяет контролировать выполнение операций с базой данных и гарантировать их

атомарность. Это особенно важно в контексте обновления учебной нагрузки, где необходимо обеспечить согласованность данных.

Наконец, стоит отметить, что интеграция базы данных в веб-приложение требует регулярного мониторинга и оптимизации. Это включает в себя анализ производительности запросов, индексацию таблиц и настройку параметров подключения. Использование инструментов мониторинга и профилирования поможет выявить узкие места и улучшить общую эффективность работы приложения.

Таким образом, интеграция базы данных в веб-приложение на Java Spring является многогранным процессом, который требует внимательного подхода к выбору технологий, проектированию структуры данных и обеспечению надежности взаимодействия с базой данных. Правильная реализация этих аспектов позволит создать эффективное и масштабируемое решение для отслеживания и оптимизации учебной нагрузки преподавателя.

ГЛАВА 2. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ К ВЕБ-ПРИЛОЖЕНИЮ ДЛЯ ОТСЛЕЖИВАНИЯ УЧЕБНОЙ НАГРУЗКИ ПРЕПОДАВАТЕЛЯ

2.1. ОПРЕДЕЛЕНИЕ ОСНОВНЫХ ВОЗМОЖНОСТЕЙ ПРИЛОЖЕНИЯ

Веб-приложение для отслеживания учебной нагрузки преподавателя должно обладать рядом функциональных возможностей, которые обеспечат его эффективность и удобство в использовании. Основные возможности приложения можно разделить на несколько ключевых категорий, каждая из которых направлена на решение конкретных задач, связанных с управлением учебной нагрузкой.

Во-первых, приложение должно предоставлять возможность регистрации и аутентификации пользователей. Преподаватели и администраторы должны

иметь возможность создавать учетные записи, а также входить в систему с использованием безопасных методов аутентификации. Это обеспечит защиту данных и позволит каждому пользователю иметь доступ только к необходимой информации.

Во-вторых, важной функцией является управление учебными курсами. Преподаватели должны иметь возможность добавлять, редактировать и удалять курсы, а также назначать их на определенные группы студентов. Это позволит поддерживать актуальность информации о курсах и обеспечит гибкость в управлении учебным процессом.

Третья ключевая возможность заключается в отслеживании учебной нагрузки. Приложение должно предоставлять инструменты для ввода и анализа данных о количестве часов, отведенных на преподавание, а также о других аспектах учебной нагрузки, таких как подготовка материалов, консультации и оценка студентов. Это позволит преподавателям более эффективно планировать свое время и ресурсы.

Кроме того, приложение должно включать функционал для генерации отчетов. Преподаватели и администраторы должны иметь возможность формировать отчеты о своей учебной нагрузке за определенные периоды, что поможет в анализе и оптимизации рабочего времени. Отчеты могут включать графики и диаграммы, что делает информацию более наглядной и доступной для восприятия.

Не менее важной является возможность взаимодействия с другими пользователями системы. Приложение должно поддерживать функции обмена сообщениями и уведомлений, что позволит преподавателям и администраторам оперативно обмениваться информацией и координировать свои действия.

Также стоит отметить необходимость интеграции с внешними системами, такими как электронные журналы и платформы для дистанционного обучения. Это обеспечит более широкий функционал и упростит процесс работы с

учебной нагрузкой, позволяя преподавателям использовать уже существующие инструменты.

Наконец, приложение должно быть интуитивно понятным и удобным в использовании. Пользовательский интерфейс должен быть разработан с учетом потребностей конечных пользователей, что позволит минимизировать время на обучение работе с системой и повысить общую продуктивность.

Таким образом, определение основных возможностей веб-приложения для отслеживания учебной нагрузки преподавателя включает в себя создание функционала для управления курсами, отслеживания нагрузки, генерации отчетов, взаимодействия пользователей и интеграции с внешними системами. Эти возможности обеспечат эффективное решение задач, связанных с управлением учебным процессом, и помогут преподавателям оптимизировать свою работу.

2.2. ПРОЕКТИРОВАНИЕ СТРУКТУРЫ БАЗЫ ДАННЫХ

Проектирование структуры базы данных является ключевым этапом в разработке веб-приложения для отслеживания учебной нагрузки преподавателя. Правильная структура базы данных обеспечивает целостность, эффективность и масштабируемость системы. В данном разделе рассматриваются основные сущности, их атрибуты и взаимосвязи, которые будут использоваться в приложении.

Первой сущностью, которую необходимо определить, является "Преподаватель". Эта сущность будет содержать информацию о каждом преподавателе, включая такие атрибуты, как уникальный идентификатор, имя, фамилия, электронная почта, телефон и кафедра. Эти данные позволят идентифицировать преподавателей и связывать их с соответствующими курсами и учебной нагрузкой.

Следующей важной сущностью является "Курс". Курс будет представлять собой учебный предмет, который ведет преподаватель. Атрибуты этой сущности могут включать уникальный идентификатор курса, название, описание, количество кредитов и семестр, в котором курс предлагается. Связь между сущностью "Курс" и "Преподаватель" будет реализована через внешний ключ, что позволит установить, какой преподаватель ведет конкретный курс.

Сущность "Студент" также играет важную роль в структуре базы данных. Она будет содержать информацию о студентах, включая уникальный идентификатор, имя, фамилию, группу и контактные данные. Связь между студентами и курсами будет реализована через промежуточную сущность "Запись на курс", которая позволит отслеживать, какие студенты записаны на какие курсы.

Для учета учебной нагрузки преподавателя необходимо создать сущность "Учебная нагрузка". Эта сущность будет содержать информацию о количестве часов, отведенных на преподавание, а также о других аспектах нагрузки, таких как подготовка материалов и консультации. Атрибуты могут включать уникальный идентификатор нагрузки, идентификатор преподавателя, идентификатор курса, количество часов и тип нагрузки (лекции, семинары, лабораторные занятия и т.д.). Связь с сущностями "Преподаватель" и "Курс" будет осуществляться через внешние ключи.

Кроме того, для поддержки взаимодействия между пользователями системы можно добавить сущность "Сообщение". Эта сущность будет содержать информацию о сообщениях, отправленных между преподавателями и администраторами. Атрибуты могут включать уникальный идентификатор сообщения, идентификатор отправителя, идентификатор получателя, текст сообщения и дату отправки.

Важно также предусмотреть механизмы для хранения истории изменений данных. Для этого можно создать сущность "Лог изменений", которая будет

фиксировать все изменения, внесенные в систему, включая информацию о том, кто и когда произвел изменения. Это обеспечит прозрачность и возможность отслеживания действий пользователей.

Визуально структура базы данных может быть представлена в виде диаграммы сущностей и связей (ER-диаграммы), которая наглядно демонстрирует взаимосвязи между сущностями и их атрибутами. Такой подход позволит разработчикам и другим заинтересованным сторонам лучше понять архитектуру базы данных и упростит процесс ее реализации.

Таким образом, проектирование структуры базы данных для веб-приложения включает в себя определение ключевых сущностей, их атрибутов и взаимосвязей. Правильная организация данных обеспечит эффективное управление учебной нагрузкой преподавателя и позволит создать надежное и масштабируемое решение для образовательного процесса.

2.3. РАЗРАБОТКА ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Разработка пользовательского интерфейса (ПИ) для веб-приложения, предназначенного для отслеживания учебной нагрузки преподавателя, является важным этапом, который напрямую влияет на удобство использования и общую эффективность системы. ПИ должен быть интуитивно понятным, функциональным и эстетически привлекательным, чтобы пользователи могли легко взаимодействовать с приложением и выполнять необходимые задачи.

Первым шагом в разработке пользовательского интерфейса является определение целевой аудитории. В данном случае основными пользователями приложения будут преподаватели и администраторы. Учитывая их потребности

и уровень технической подготовки, интерфейс должен быть простым и доступным, с четкой навигацией и минимальным количеством шагов для выполнения основных операций.

Структура интерфейса может быть организована по принципу "панели навигации", которая будет содержать основные разделы приложения, такие как "Главная", "Курсы", "Учебная нагрузка", "Студенты", "Отчеты" и "Настройки". Панель навигации должна быть расположена в верхней части или сбоку экрана, что обеспечивает легкий доступ к различным функциям приложения.

На главной странице пользователи должны видеть сводную информацию о своей учебной нагрузке, предстоящих занятиях и уведомлениях. Это может быть реализовано в виде дашборда, который будет включать графики, диаграммы и списки, позволяющие быстро оценить текущее состояние дел. Важно, чтобы информация была представлена в наглядной и понятной форме, что поможет пользователям быстро ориентироваться в данных.

Раздел "Курсы" должен предоставлять возможность добавления, редактирования и удаления курсов. Интерфейс для работы с курсами может включать таблицу с основными атрибутами курса, такими как название, преподаватель и семестр, а также кнопки для выполнения операций. Форма добавления или редактирования курса должна быть простой и содержать только необходимые поля, чтобы избежать перегрузки пользователя информацией.

В разделе "Учебная нагрузка" пользователи должны иметь возможность вводить данные о количестве часов, отведенных на преподавание, а также просматривать и анализировать свою нагрузку. Здесь можно использовать графические элементы, такие как диаграммы и графики, для визуализации данных, что поможет преподавателям лучше понять распределение своей нагрузки.

Раздел "Студенты" должен предоставлять информацию о студентах, записанных на курсы, а также возможность управления записями. Интерфейс может

включать фильтры и поиск, что упростит навигацию по списку студентов. Также важно предусмотреть возможность отправки уведомлений или сообщений студентам через интерфейс.

Для генерации отчетов можно создать отдельный раздел, где пользователи смогут выбирать параметры для формирования отчетов о своей учебной нагрузке. Интерфейс должен быть интуитивно понятным, с возможностью предварительного просмотра отчетов перед их созданием.

Не менее важным аспектом является адаптивность интерфейса. Веб-приложение должно корректно отображаться на различных устройствах, включая настольные компьютеры, планшеты и смартфоны. Это обеспечит удобство использования приложения в любых условиях и позволит преподавателям работать с системой в любое время и в любом месте.

Наконец, стоит уделить внимание эстетическому оформлению интерфейса. Использование единой цветовой схемы, шрифтов и иконок создаст гармоничный и профессиональный вид приложения. Также важно обеспечить доступность интерфейса для пользователей с ограниченными возможностями, что может включать поддержку экранных читалок и возможность изменения размера шрифта.

Таким образом, разработка пользовательского интерфейса для веб-приложения включает в себя создание интуитивно понятной структуры, удобной навигации, визуализации данных и адаптивного дизайна. Эти аспекты обеспечат комфортное взаимодействие пользователей с приложением и помогут эффективно управлять учебной нагрузкой преподавателя.

ГЛАВА 3. АРХИТЕКТУРА И СТРУКТУРА ВЕБ-ПРИЛОЖЕНИЯ НА JAVA SPRING

3.1. МОДЕЛЬ-ПРЕДСТАВЛЕНИЕ-КОНТРОЛЛЕР (MVC) ПАТТЕРН

Модель-Представление-Контроллер (MVC) является одним из наиболее распространенных архитектурных паттернов, используемых в разработке веб-приложений, включая приложения на Java Spring. Этот паттерн разделяет приложение на три основных компонента: модель, представление и контроллер, что способствует улучшению организации кода, его тестируемости и поддерживаемости.

Модель представляет собой компонент, который отвечает за управление данными и бизнес-логикой приложения. В контексте веб-приложения для отслеживания учебной нагрузки преподавателя модель будет включать в себя классы, которые описывают основные сущности, такие как "Преподаватель", "Курс", "Студент" и "Учебная нагрузка". Эти классы будут содержать атрибуты, методы для работы с данными и взаимодействия с базой данных через репозитории, реализованные с использованием Spring Data JPA. Модель также может включать в себя бизнес-логику, такую как расчеты учебной нагрузки или валидацию данных.

Представление отвечает за отображение данных пользователю и взаимодействие с ним. В веб-приложении на Java Spring представление обычно реализуется с использованием шаблонов, таких как Thymeleaf или JSP. Эти шаблоны позволяют динамически генерировать HTML-страницы на основе данных, полученных от модели. Представление должно быть интуитивно понятным и удобным для пользователя, обеспечивая визуализацию информации о курсах, учебной нагрузке и других аспектах приложения. Важно, чтобы

представление было адаптивным и корректно отображалось на различных устройствах.

Контроллер служит связующим звеном между моделью и представлением. Он обрабатывает входящие запросы от пользователя, взаимодействует с моделью для получения или изменения данных и передает результаты в представление для отображения. В веб-приложении на Java Spring контроллеры реализуются с использованием аннотаций, таких как @Controller и @RequestMapping. Контроллеры обрабатывают различные HTTP-запросы, такие как GET, POST, PUT и DELETE, и определяют, какие действия должны быть выполнены в ответ на эти запросы.

Одним из ключевых преимуществ использования паттерна MVC является возможность разделения ответственности между компонентами. Это позволяет разработчикам работать над различными аспектами приложения независимо друг от друга. Например, дизайнеры могут сосредоточиться на создании пользовательского интерфейса, в то время как разработчики могут работать над бизнес-логикой и взаимодействием с базой данных. Кроме того, такое разделение облегчает тестирование, так как каждый компонент можно тестировать отдельно.

В контексте веб-приложения для отслеживания учебной нагрузки преподавателя, использование паттерна MVC позволяет создать четкую архитектуру, которая упрощает дальнейшую разработку и поддержку приложения. Например, если в будущем потребуется изменить логику расчета учебной нагрузки, это можно сделать в модели, не затрагивая представление или контроллер. Аналогично, изменения в пользовательском интерфейсе могут быть реализованы в представлении без необходимости изменения бизнес-логики.

Таким образом, паттерн Модель-Представление-Контроллер (MVC) является основой архитектуры веб-приложения на Java Spring. Он обеспечивает четкое

разделение ответственности между компонентами, что способствует улучшению организации кода, его тестируемости и поддерживаемости. Использование этого паттерна позволяет создать эффективное и масштабируемое решение для отслеживания учебной нагрузки преподавателя, обеспечивая удобство и функциональность для конечных пользователей.

3.2. КОНФИГУРАЦИЯ И НАСТРОЙКА ПРИЛОЖЕНИЯ

Конфигурация и настройка веб-приложения на Java Spring являются важными этапами, которые обеспечивают правильную работу приложения, его интеграцию с необходимыми компонентами и настройку параметров, влияющих на производительность и безопасность. В данном разделе рассматриваются ключевые аспекты конфигурации и настройки приложения для отслеживания учебной нагрузки преподавателя.

Первым шагом в конфигурации приложения является настройка зависимостей. В современных проектах на Java Spring обычно используется система управления зависимостями Maven или Gradle. В файле конфигурации (pom.xml для Maven или build.gradle для Gradle) необходимо указать все необходимые зависимости, такие как Spring Boot, Spring Data JPA, драйвер базы данных (например, PostgreSQL или MySQL), а также библиотеки для работы с шаблонами (например, Thymeleaf). Это позволит автоматически загружать и управлять версиями библиотек, необходимых для работы приложения.

Следующим этапом является настройка подключения к базе данных. В приложении на Java Spring это обычно делается через файл application.properties или application.yml, где указываются параметры подключения, такие как URL базы данных, имя пользователя и пароль. Например, для PostgreSQL это может выглядеть следующим образом:

```
spring:
```

```yaml
datasource:
  url: jdbc:postgresql://localhost:5432/myDataBase
  username: Polyakov
  password: tiger123
  driverClassName: org.postgresql.Driver
jpa:
  hibernate:
    ddl-auto: update
  show-sql: true
logging:
level:
  root: INFO # Уровень логирования для корневого логгера
  org.springframework: DEBUG # Уровень логирования для Spring Framework
   com.yourpackage: DEBUG # Уровень логирования для вашего пакета (замените на ваш пакет)
```

Эти настройки обеспечивают подключение к базе данных и конфигурируют Hibernate для автоматического обновления схемы базы данных в зависимости от изменений в моделях.

Кроме того, важно настроить параметры безопасности приложения. Для этого можно использовать Spring Security, который предоставляет мощные инструменты для аутентификации и авторизации пользователей. В конфигурации безопасности необходимо определить, какие URL-адреса требуют аутентификации, а какие доступны для всех пользователей. Также следует настроить механизмы аутентификации, такие как форма входа или OAuth2, в зависимости от требований проекта.

Настройка кэширования и управления сессиями также является важным аспектом конфигурации. Spring предоставляет возможности для кэширования данных, что может значительно повысить производительность приложения. Настройка кэширования может быть выполнена через аннотации, такие как @Cacheable и @CacheEvict, которые позволяют управлять кэшированием на уровне методов.

Для обеспечения удобства работы с приложением и улучшения пользовательского опыта можно настроить международализацию (i18n). Это позволит поддерживать несколько языков интерфейса, что особенно важно в образовательных учреждениях с многоязычной аудиторией. Настройка международализации включает в себя создание файлов с переводами и конфигурацию локализации в приложении.

Также стоит обратить внимание на логирование. Spring Boot предоставляет встроенные возможности для логирования, которые можно настроить в файле application.properties. Настройка уровня логирования (например, DEBUG, INFO, WARN, ERROR) и формата сообщений поможет в отладке приложения и мониторинге его работы.

Наконец, важно предусмотреть механизмы для тестирования приложения. Настройка тестовой среды, включая использование JUnit и Mockito, позволит разработчикам проверять функциональность приложения и выявлять ошибки на ранних этапах разработки.

Таким образом, конфигурация и настройка веб-приложения на Java Spring включают в себя настройку зависимостей, подключение к базе данных, безопасность, кэширование, международализацию, логирование и тестирование. Правильная настройка этих аспектов обеспечит надежную и эффективную работу приложения для отслеживания учебной нагрузки преподавателя, а также создаст удобные условия для пользователей.

3.3. ОБРАБОТКА И ХРАНЕНИЕ ДАННЫХ

Обработка и хранение данных являются ключевыми аспектами разработки веб-приложения на Java Spring, особенно в контексте приложения для отслеживания учебной нагрузки преподавателя. Эти процессы включают в себя взаимодействие с базой данных, управление данными, их валидацию и обеспечение целостности. В данном разделе рассматриваются основные подходы к обработке и хранению данных в приложении.

Первым шагом в обработке данных является создание моделей, которые представляют собой классы, соответствующие сущностям базы данных. Каждая модель содержит атрибуты, которые отражают поля таблиц в базе данных, а также методы для работы с этими данными. Важно использовать аннотации JPA (Java Persistence API) для определения связи между моделями и таблицами, а также для указания типов данных и ограничений. Например, аннотация @Entity указывает, что класс является сущностью, а аннотация @Table позволяет задать имя таблицы в базе данных.

Для взаимодействия с базой данных в приложении используется Spring Data JPA, который упрощает работу с репозиториями. Репозитории представляют собой интерфейсы, которые обеспечивают доступ к данным и позволяют выполнять операции CRUD (создание, чтение, обновление, удаление). Разработчики могут создавать интерфейсы, расширяющие JpaRepository, что позволяет автоматически получать реализацию основных методов для работы с данными. Например, репозиторий для сущности "Преподаватель" может выглядеть следующим образом:

```
import AlexeyPolyakov.MyFinalWork.DataBase.UserEntity;

import org.springframework.data.jpa.repository.JpaRepository;
```

```
public interface UserRepository extends JpaRepository<UserEntity, Long> {

   UserEntity findByEmail(String email);

}
```

Обработка данных также включает в себя валидацию входящих данных. Веб-приложение должно обеспечивать проверку корректности данных, вводимых пользователями, чтобы избежать ошибок и обеспечить целостность информации. Для этого можно использовать аннотации валидации, такие как @NotNull, @Size и @Email, которые позволяют автоматически проверять данные на этапе их ввода. Валидация может быть реализована как на уровне модели, так и на уровне контроллера, что обеспечивает дополнительный уровень защиты.

После обработки данных и их валидации необходимо обеспечить их сохранение в базе данных. Spring Data JPA предоставляет механизмы для выполнения операций сохранения и обновления данных. Например, метод save() репозитория позволяет сохранять или обновлять сущности в базе данных. Важно также учитывать управление транзакциями, чтобы гарантировать атомарность операций. Spring поддерживает как программные, так и декларативные транзакции, что позволяет разработчикам выбирать наиболее подходящий подход для их приложения.

Кроме того, необходимо предусмотреть механизмы для обработки ошибок, возникающих при работе с данными. Это может включать в себя обработку исключений, связанных с доступом к базе данных, а также валидацией данных. Spring предоставляет возможности для глобальной обработки исключений с помощью аннотации @ControllerAdvice, что позволяет централизовать логику

обработки ошибок и возвращать пользователям понятные сообщения об ошибках.

Хранение данных в базе данных также требует учета вопросов безопасности. Необходимо обеспечить защиту данных от несанкционированного доступа и утечек. Это может включать в себя шифрование конфиденциальной информации, такой как пароли, а также использование безопасных соединений (например, HTTPS) для передачи данных между клиентом и сервером.

Наконец, стоит отметить важность мониторинга и оптимизации работы с данными. Это может включать в себя анализ производительности запросов к базе данных, использование индексов для ускорения поиска и регулярное резервное копирование данных. Инструменты мониторинга, такие как Spring Actuator, могут помочь в отслеживании состояния приложения и выявлении узких мест.

Таким образом, обработка и хранение данных в веб-приложении на Java Spring включают в себя создание моделей, взаимодействие с базой данных через репозитории, валидацию данных, управление транзакциями, обработку ошибок и обеспечение безопасности. Эти аспекты являются критически важными для создания надежного и эффективного приложения для отслеживания учебной нагрузки преподавателя, обеспечивая целостность и безопасность данных.

ГЛАВА 4. РЕАЛИЗАЦИЯ ФУНКЦИОНАЛА ОТСЛЕЖИВАНИЯ И ОПТИМИЗАЦИИ УЧЕБНОЙ НАГРУЗКИ ПРЕПОДАВАТЕЛЯ

4.1. РАЗРАБОТКА МОДУЛЯ ОТСЛЕЖИВАНИЯ УЧЕБНОЙ НАГРУЗКИ

Разработка модуля отслеживания учебной нагрузки преподавателя является важной частью веб-приложения, предназначенного для управления образовательным процессом. Этот модуль должен обеспечивать сбор, обработку и представление данных о нагрузке преподавателей, что позволит им

эффективно планировать свое время и ресурсы. В данном разделе рассматриваются ключевые аспекты реализации данного функционала.

Первым шагом в разработке модуля является определение структуры данных, необходимых для отслеживания учебной нагрузки. Важно создать модель, которая будет представлять учебную нагрузку преподавателя. Эта модель может включать такие атрибуты, как уникальный идентификатор нагрузки, идентификатор преподавателя, идентификатор курса, количество часов, отведенных на лекции, семинары и другие виды деятельности, а также тип нагрузки (например, преподавание, подготовка материалов, консультации).

Для хранения информации о нагрузке можно использовать сущность "Учебная нагрузка", которая будет связана с сущностями "Преподаватель" и "Курс". Связь между этими сущностями позволит легко получать информацию о том, какая нагрузка назначена конкретному преподавателю и на какие курсы.

Следующим этапом является создание репозитория для работы с данными о учебной нагрузке. Используя Spring Data JPA, можно создать интерфейс, который будет обеспечивать доступ к данным и выполнять операции CRUD. Например, репозиторий может включать методы для поиска нагрузки по преподавателю, курсу или типу нагрузки.

После создания модели и репозитория необходимо реализовать бизнес-логику для обработки данных о нагрузке. Это может включать в себя методы для добавления новой нагрузки, обновления существующей информации и удаления ненужных записей. Важно также предусмотреть валидацию данных на этапе их ввода, чтобы избежать ошибок и обеспечить целостность информации.

Для удобства пользователей необходимо разработать интерфейс, который позволит преподавателям легко вводить и редактировать данные о своей учебной нагрузке. Это может быть реализовано через веб-формы, где пользователи смогут выбирать курсы, вводить количество часов и указывать

тип нагрузки. Интерфейс должен быть интуитивно понятным и адаптивным, чтобы обеспечить комфортное взаимодействие на различных устройствах.

Кроме того, модуль отслеживания учебной нагрузки должен включать функционал для генерации отчетов. Преподаватели должны иметь возможность формировать отчеты о своей нагрузке за определенные периоды, что поможет им анализировать распределение времени и оптимизировать свою работу. Отчеты могут включать графики и диаграммы, что делает информацию более наглядной и доступной для восприятия.

Для повышения эффективности работы модуля можно реализовать механизмы уведомлений. Например, преподаватели могут получать уведомления о предстоящих занятиях, изменениях в расписании или необходимости обновления данных о нагрузке. Это поможет им оставаться в курсе событий и своевременно реагировать на изменения.

Наконец, важно предусмотреть возможность интеграции модуля с другими компонентами приложения, такими как управление курсами и студентами. Это обеспечит более полное представление о учебном процессе и позволит преподавателям лучше планировать свою работу.

Таким образом, разработка модуля отслеживания учебной нагрузки преподавателя включает в себя создание модели данных, репозитория, бизнес-логики, пользовательского интерфейса, функционала для генерации отчетов и механизмов уведомлений. Эти аспекты обеспечат эффективное управление учебной нагрузкой и помогут преподавателям оптимизировать свое время и ресурсы, что в конечном итоге приведет к улучшению качества образовательного процесса.

4.2. РЕАЛИЗАЦИЯ ФУНКЦИОНАЛА ОПТИМИЗАЦИИ РАСПИСАНИЯ

Реализация функционала оптимизации расписания в веб-приложении для отслеживания учебной нагрузки преподавателя является важным шагом к повышению эффективности образовательного процесса. Этот функционал позволяет преподавателям и администраторам более рационально распределять учебные часы, избегать конфликтов в расписании и обеспечивать сбалансированную нагрузку. В данном разделе рассматриваются ключевые аспекты разработки данного функционала.

Первым шагом в реализации функционала оптимизации расписания является анализ текущих данных о расписании и учебной нагрузке. Для этого необходимо создать модель, которая будет представлять расписание занятий. Эта модель может включать такие атрибуты, как уникальный идентификатор занятия, идентификатор курса, идентификатор преподавателя, время начала и окончания занятия, а также аудиторию. Связь между расписанием и сущностями "Курс" и "Преподаватель" позволит легко получать информацию о том, какие занятия назначены конкретным преподавателям и на какие курсы.

Следующим этапом является создание репозитория для работы с данными о расписании. Используя Spring Data JPA, можно создать интерфейс, который будет обеспечивать доступ к данным и выполнять операции CRUD. Репозиторий может включать методы для поиска расписания по преподавателю, курсу или времени, что упростит работу с данными.

Для оптимизации расписания необходимо разработать алгоритмы, которые будут анализировать текущую нагрузку и предлагать изменения. Это может включать в себя:

1. Анализ конфликтов: Алгоритм должен проверять, есть ли пересечения в расписании для одного и того же преподавателя или группы студентов. Если такие конфликты обнаружены, система должна предлагать альтернативные временные слоты.

2. Балансировка нагрузки: Алгоритм может анализировать распределение учебной нагрузки между преподавателями и предлагать изменения, чтобы избежать перегрузки отдельных преподавателей. Это может включать в себя перераспределение часов между курсами или изменение времени занятий.

3. Оптимизация использования аудиторий: Система может анализировать доступность аудиторий и предлагать оптимальные варианты размещения занятий, чтобы минимизировать время простоя и обеспечить максимальную эффективность использования ресурсов.

4. Учет предпочтений: Важно учитывать предпочтения преподавателей и студентов при составлении расписания. Система может предоставлять возможность пользователям указывать свои предпочтения по времени и дням занятий, что поможет создать более удобное расписание.

После разработки алгоритмов необходимо реализовать пользовательский интерфейс, который позволит преподавателям и администраторам взаимодействовать с функционалом оптимизации расписания. Интерфейс должен предоставлять возможность просмотра текущего расписания, а также предлагать варианты оптимизации. Пользователи должны иметь возможность легко вносить изменения и подтверждать новые варианты расписания.

Кроме того, важно предусмотреть механизмы уведомлений, которые будут информировать преподавателей и студентов о изменениях в расписании. Это может включать в себя отправку уведомлений по электронной почте или через встроенные сообщения в приложении, что поможет пользователям оставаться в курсе изменений.

Для повышения эффективности работы функционала оптимизации расписания можно интегрировать его с другими модулями приложения, такими как отслеживание учебной нагрузки и управление курсами. Это обеспечит более

полное представление о учебном процессе и позволит принимать более обоснованные решения при оптимизации расписания.

Наконец, стоит отметить важность тестирования и мониторинга работы функционала. Регулярное тестирование алгоритмов оптимизации и анализ их эффективности помогут выявить узкие места и улучшить качество работы системы.

Таким образом, реализация функционала оптимизации расписания включает в себя создание модели и репозитория для работы с данными о расписании, разработку алгоритмов для анализа и оптимизации нагрузки, создание пользовательского интерфейса, механизмы уведомлений и интеграцию с другими модулями приложения. Эти аспекты обеспечат эффективное управление расписанием и помогут преподавателям и администраторам оптимизировать учебный процесс, что в конечном итоге приведет к улучшению качества образования.

4.3. ТЕСТИРОВАНИЕ И ОТЛАДКА ФУНКЦИОНАЛА

Тестирование и отладка функционала веб-приложения для отслеживания и оптимизации учебной нагрузки преподавателя являются критически важными этапами разработки, которые обеспечивают надежность, производительность и соответствие требованиям пользователей. В данном разделе рассматриваются функционала приложения.

Тестирование функционала можно разделить на несколько видов:

- Модульное тестирование: Этот вид тестирования направлен на проверку отдельных компонентов приложения, таких как модели, репозитории и сервисы. Модульные тесты позволяют убедиться, что каждая часть приложения работает корректно и выполняет свои функции. В Java Spring для модульного тестирования часто используются библиотеки JUnit и

Mockito, которые позволяют создавать тесты и имитировать поведение зависимостей.

- Интеграционное тестирование: Интеграционные тесты проверяют взаимодействие между различными компонентами приложения, такими как контроллеры, сервисы и репозитории. Эти тесты помогают выявить проблемы, возникающие при взаимодействии различных частей системы. В Spring можно использовать аннотацию @SpringBootTest, которая загружает контекст приложения и позволяет тестировать его в реальных условиях.

- Функциональное тестирование: Этот вид тестирования направлен на проверку функциональности приложения с точки зрения пользователя. Функциональные тесты проверяют, выполняются ли все бизнес-требования и сценарии использования. Для автоматизации функционального тестирования можно использовать инструменты, такие как Selenium или Cucumber, которые позволяют имитировать действия пользователя в веб-интерфейсе.

- Тестирование производительности: Тестирование производительности позволяет оценить, как приложение справляется с нагрузкой. Это может включать в себя стресс-тестирование, нагрузочное тестирование и тестирование на устойчивость. Инструменты, такие как JMeter или Gatling, могут быть использованы для симуляции большого количества пользователей и анализа производительности приложения.

Процесс тестирования включает в себя несколько этапов:

- Планирование тестирования: На этом этапе определяются цели тестирования, виды тестов, которые будут проводиться, и ресурсы, необходимые для их выполнения. Также разрабатывается тестовая документация, включая тестовые сценарии и критерии приемки.

- Разработка тестов: На этом этапе создаются тестовые случаи и сценарии, которые будут использоваться для проверки функционала. Важно, чтобы тесты были четко документированы и легко воспроизводимы.

- Выполнение тестов: Тесты выполняются в соответствии с разработанными сценариями. Результаты тестирования фиксируются, и выявленные ошибки и проблемы документируются.

- Анализ результатов: После выполнения тестов результаты анализируются, и принимаются решения о необходимости исправления ошибок или доработки функционала. Важно также оценить, насколько тесты покрывают функциональность приложения.

- Отладка: На этапе отладки разработчики работают над исправлением выявленных ошибок. Это может включать в себя анализ логов, использование отладчиков и профилировщиков для выявления узких мест и проблем в коде.

Для эффективного тестирования и отладки веб-приложения можно использовал различные инструменты:

- JUnit: Библиотека для модульного тестирования в Java, позволяющая создавать и выполнять тесты.

- Postman: Инструмент для тестирования API, который позволяет отправлять запросы к серверу и проверять ответы.

Тестирование и отладка функционала веб-приложения для отслеживания и оптимизации учебной нагрузки преподавателя являются важными этапами, которые обеспечивают надежность и качество приложения. Использование различных видов тестирования, четкий процесс тестирования и применение современных инструментов позволяют выявлять и исправлять ошибки на ранних этапах разработки, что в конечном итоге приводит к созданию более качественного и эффективного продукта. Регулярное тестирование и мониторинг работы приложения также помогут поддерживать его актуальность и соответствие требованиям пользователей в будущем.

Глава 5. Тестирование, оптимизация и дальнейшее развитие веб-приложения

В процессе разработки веб-приложения для отслеживания и оптимизации учебной нагрузки преподавателя на платформе Java Spring особое внимание было уделено тестированию, оптимизации и планированию дальнейшего развития системы. Эти этапы являются критически важными для обеспечения надежности, производительности и соответствия приложения требованиям пользователей.

Тестирование приложения началось с модульного тестирования, которое позволило проверить отдельные компоненты системы на корректность их работы. Каждый модуль был протестирован в изоляции, что дало возможность выявить и устранить ошибки на ранних стадиях разработки. Модульные тесты охватывали ключевые функции, такие как управление учебной нагрузкой, обработка данных пользователей и взаимодействие с базой данных. Это обеспечивало уверенность в том, что каждый элемент системы функционирует должным образом и соответствует заданным спецификациям.

После завершения модульного тестирования было проведено интеграционное тестирование, целью которого было проверить взаимодействие между различными модулями приложения. Этот этап позволил выявить проблемы, возникающие при совместной работе компонентов, и убедиться в том, что данные корректно передаются и обрабатываются на всех уровнях системы. Интеграционное тестирование также включало проверку взаимодействия с внешними сервисами и API, что является важным аспектом для обеспечения функциональности приложения в реальных условиях.

Оптимизация веб-приложения была направлена на улучшение его производительности и отзывчивости. В ходе анализа производительности были выявлены узкие места, которые могли негативно сказаться на пользовательском опыте. Оптимизация запросов к базе данных, использование кэширования и

улучшение алгоритмов обработки данных стали ключевыми мерами, направленными на повышение эффективности работы приложения. Также была проведена работа по улучшению пользовательского интерфейса, что способствовало более интуитивному взаимодействию пользователей с системой.

Дальнейшее развитие веб-приложения предполагает регулярное обновление функционала в соответствии с изменяющимися потребностями пользователей и образовательной среды. Планируется внедрение новых возможностей, таких как расширенные аналитические инструменты для преподавателей, интеграция с другими образовательными платформами и улучшение системы отчетности. Также важным направлением является сбор и анализ обратной связи от пользователей, что позволит оперативно реагировать на их запросы и улучшать качество предоставляемых услуг.

Таким образом, тестирование, оптимизация и планирование дальнейшего развития веб-приложения являются неотъемлемыми частями процесса разработки, которые способствуют созданию надежного и эффективного инструмента для отслеживания и оптимизации учебной нагрузки преподавателя.

ПЛАНИРОВАНИЕ ДАЛЬНЕЙШИХ УЛУЧШЕНИЙ И РАЗВИТИЕ ФУНКЦИОНАЛА

Планирование дальнейших улучшений и развитие функционала веб-приложения для отслеживания и оптимизации учебной нагрузки преподавателя является важным этапом, который обеспечивает его актуальность и соответствие потребностям пользователей. В условиях динамично

меняющейся образовательной среды необходимо постоянно адаптировать приложение, добавляя новые функции и улучшая существующие.

Одним из ключевых направлений дальнейшего развития является расширение аналитических возможностей приложения. Внедрение более сложных алгоритмов анализа данных позволит преподавателям не только отслеживать свою учебную нагрузку, но и получать рекомендации по оптимизации времени и ресурсов. Например, можно разработать инструменты для прогнозирования нагрузки на основе исторических данных, что поможет преподавателям более эффективно планировать свои занятия и распределять время.

Также планируется интеграция с другими образовательными платформами и системами управления обучением. Это позволит создать единое информационное пространство, где преподаватели смогут получать доступ к необходимым данным и инструментам в одном месте. Интеграция с системами видеоконференций, электронными библиотеками и другими ресурсами значительно упростит процесс подготовки и проведения занятий.

Важным аспектом дальнейшего развития является улучшение пользовательского интерфейса. Удобство и интуитивность интерфейса играют ключевую роль в восприятии приложения пользователями. Планируется провести исследование пользовательского опыта, чтобы выявить проблемные зоны и области для улучшения. На основе полученных данных будут разработаны новые макеты и прототипы, которые сделают взаимодействие с приложением более комфортным и эффективным.

Кроме того, стоит обратить внимание на мобильную версию приложения. В условиях современного мира, где мобильные устройства становятся основным средством доступа к информации, создание адаптивного дизайна и функционала для мобильных платформ позволит расширить аудиторию пользователей и повысить доступность приложения.

Не менее важным направлением является работа с обратной связью от пользователей. Регулярный сбор отзывов и предложений поможет выявить актуальные потребности и ожидания преподавателей, что позволит оперативно реагировать на изменения и вносить необходимые коррективы в функционал приложения. Создание системы поддержки пользователей, включая обучающие материалы и FAQ, также будет способствовать более эффективному использованию приложения.

Таким образом, планирование дальнейших улучшений и развитие функционала веб-приложения должны основываться на анализе потребностей пользователей, современных тенденциях в образовании и технологиях. Это позволит создать инструмент, который будет не только соответствовать текущим требованиям, но и предвосхищать ожидания пользователей, обеспечивая тем самым его долгосрочную актуальность и успешность.

Приложения

В **приложения** обычно входят артефакты, получившиеся в процессе создания проекта:

1. Объёмные графики и таблицы, которые не помещаются на лист А4.
2. Длинные математические формулы и расчёты по ним.
3. Характеристики аппаратуры, которая использовалась для проведения исследования.
4. Авторские методики.
5. Вспомогательный материал: тесты, карточки, схемы, рисунки.
6. Материалы, полученные на предприятии: отчёты, прочие документы.

ЗАКЛЮЧЕНИЕ

В заключение данной работы на тему «Веб-приложение для отслеживания и оптимизации учебной нагрузки преподавателя Java Spring» можно подвести итоги проделанной работы, выделив ключевые аспекты, которые были рассмотрены в ходе исследования и разработки. Прежде всего, стоит отметить, что создание веб-приложения на платформе Java Spring представляет собой актуальную задачу, учитывая растущие требования к повышению эффективности образовательного процесса. В условиях современного образования, где преподаватели сталкиваются с необходимостью управлять значительными объемами информации, подобные инструменты становятся незаменимыми помощниками.

В процессе работы над курсовой были изучены технологии разработки веб-приложений на Java Spring, которые являются основой для реализации нашего проекта. Java Spring предоставляет мощный и гибкий фреймворк, который позволяет создавать масштабируемые и надежные приложения. Основные компоненты Spring, такие как Spring MVC, Spring Boot и Spring Data,

обеспечивают удобство разработки, упрощая создание RESTful API, управление базами данных и обработку запросов. Важно отметить, что использование Spring позволяет разработчикам сосредоточиться на логике приложения, минимизируя время на решение рутинных задач, таких как конфигурация и интеграция различных модулей.

Функциональные требования к веб-приложению были детально проработаны и включали в себя ключевые аспекты, необходимые для эффективного отслеживания учебной нагрузки преподавателя. В процессе анализа требований мы определили, что приложение должно обеспечивать возможность ввода и редактирования учебной нагрузки, мониторинга выполнения учебных планов, а также генерации отчетов, что позволит преподавателям более эффективно планировать свое время и ресурсы. Также было важно обеспечить интуитивно понятный интерфейс, чтобы пользователи могли легко взаимодействовать с системой и получать необходимую информацию без дополнительных усилий.

Архитектура и структура веб-приложения были разработаны с учетом принципов модульности и масштабируемости. Мы применили архитектурный подход, основанный на разделении приложения на несколько слоев, таких как слой представления, бизнес-логики и доступа к данным. Это позволило не только упростить процесс разработки, но и обеспечить легкость в дальнейшем обслуживании и расширении функционала. Использование паттернов проектирования, таких как MVC (Model-View-Controller), способствовало четкому разделению ответственности между компонентами, что является важным аспектом для поддерживаемости кода.

Реализация функционала отслеживания и оптимизации учебной нагрузки преподавателя стала центральным элементом нашей работы. Мы разработали механизм, который позволяет преподавателям вводить данные о своей учебной нагрузке, а также анализировать их с помощью различных метрик. Важно отметить, что оптимизация учебной нагрузки является многогранной задачей, требующей не только учета текущих данных, но и анализа исторической

информации. Для этого была реализована система отчетности, которая позволяет пользователям визуализировать данные и делать выводы о своей работе. Также мы внедрили функционал, позволяющий преподавателям получать рекомендации по оптимизации своей нагрузки на основе анализа данных.

Тестирование, оптимизация и дальнейшее развитие веб-приложения стали завершающими этапами работы. Мы провели комплексное тестирование, которое включало как функциональные, так и нагрузочные испытания. Это позволило выявить и устранить потенциальные проблемы, а также оптимизировать производительность приложения. В результате мы смогли достичь стабильной работы системы даже при увеличении нагрузки, что подтверждает высокую надежность разработанного решения. Важно отметить, что тестирование является неотъемлемой частью процесса разработки, и его результаты позволяют не только улучшить текущее состояние приложения, но и заложить основу для его дальнейшего развития.

Дальнейшее развитие веб-приложения предполагает внедрение новых функций и возможностей, которые могут повысить его эффективность и удобство использования. Мы рассматриваем возможность интеграции с другими образовательными платформами, что позволит расширить функционал и улучшить взаимодействие пользователей с системой. Кроме того, планируется внедрение искусственного интеллекта для более глубокого анализа данных и предоставления персонализированных рекомендаций преподавателям. Это позволит не только оптимизировать учебную нагрузку, но и повысить качество образовательного процесса в целом.

Таким образом, в ходе работы над курсовой мы смогли создать полноценное веб-приложение для отслеживания и оптимизации учебной нагрузки преподавателя на базе Java Spring. Мы проанализировали ключевые технологии, разработали функциональные требования, спроектировали архитектуру и

реализовали необходимые функции. Тестирование и оптимизация стали завершающими этапами, которые подтвердили надежность и эффективность разработанного решения. В дальнейшем, с учетом полученного опыта и анализа потребностей пользователей, мы планируем продолжить работу над проектом, внедряя новые функции и улучшая взаимодействие с пользователями. В результате, наше веб-приложение станет не только инструментом для управления учебной нагрузкой, но и важным помощником для преподавателей в их профессиональной деятельности.

СПИСОК ЛИТЕРАТУРЫ

1. Tofiq Y. et al. Tələbələrin biliyinin analizi üsullarının və proqram təminatının işlənməsi : дис. – Azərbaycan Texniki Universiteti, 2023. URL: http://openaccess.aztu.edu.az/xmlui/bitstream/handle/123456789/122/Tələbələrin biliyinin analizi üsullarının və proqram təminatının işlənməsi.pdf?sequence=1&isAllowed=y (дата обращения: 05.11.2024).

2. Александров К. В. Разработка инструментальных средств для анализа информационной значимости компонентов состояния здоровья и показателей физической подготовки с учетом возрастных особенностей. – 2021. URL: https://earchive.tpu.ru/handle/11683/67259 (дата обращения: 05.11.2024).

3. Анвар Шухратович Абдуллаев, Тохиржон Азаматович Саъдуллаев РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ РЕГИСТРАЦИИ ПАЦИЕНТОВ ГЛАЗНЫХ КЛИНИК // Academic research in educational sciences. 2023. №5. URL: https://cyberleninka.ru/article/n/razrabotki-veb-prilozheniya-dlya-registratsii-patsientov-glaznyh-klinik (дата обращения: 05.11.2024).

4. Антончик Д. И. Закономерности прогорания пористых огнепреградителей. – 2016. URL: https://www.bsuir.by/m/12_100229_1_106529.pdf (дата обращения: 05.11.2024).

5. Антончик Д. И. ека БГУИР. URL: https://libeldoc.bsuir.by/bitstream/123456789/44937/1/52_konf_fkp_2016.pdf (дата обращения: 05.11.2024).

6. Аршанский Е. Я. и др. ББК 74.480. 278я431+ 74.483 (4Беи) я431 М75. – 2023. URL: https://elibrary.ru/download/elibrary_53867558_33885323.pdf (дата обращения: 05.11.2024).

7. Байделюк Е. А. Разработка системы «HR-automation». – 2023. URL: https://earchive.tpu.ru/handle/11683/76382 (дата обращения: 05.11.2024).

8. Бойко И. Д. Проектирование и реализация микросервисной архитектуры веб-платформы для обучения иностранным языкам. – 2021. URL: https://earchive.tpu.ru/handle/11683/67205 (дата обращения: 05.11.2024).

9. Вотинов Максим Валерьевич Особенности построения Web-приложений информационного обеспечения систем автоматического управления // Вестник АГТУ. Серия: Управление, вычислительная техника и информатика. 2017. №3. URL: https://cyberleninka.ru/article/n/osobennosti-postroeniya-web-prilozheniy-informatsionnogo-obespecheniya-sistem-avtomaticheskogo-upravleniya (дата обращения: 05.11.2024).

10. Головин Г. И., Аксенов С. Г. Анализ обеспечения пожарной безопасности на летно-испытательных станциях, предприятий авиационной промышленности // Студенческий форум. – 2021. – Т. 33. – №. 169. – С. 31-33. URL: https://nauchforum.ru/archive/studjournal/33(169).pdf#page=32 (дата обращения: 05.11.2024).

11. Григорьева Ю. Г., Черкашина М. А. НЕОБХОДИМОСТЬ ПРОВЕДЕНИЯ РЕИНЖИНИРИНГА В ПРОИЗВОДСТВЕННЫХ ПРЕДПРИЯТИЯХ ПРИ ВНЕДРЕНИИ ТИПОВЫХ ИНФОРМАЦИОННЫХ СИСТЕМ // Инжиниринг предприятий и управление знаниями (ИП&УЗ-2017). – 2017. – С. 64. URL:

http://conf-eekm.ru/wp-content/uploads/2016/01/molod-2017.pdf#page=64 (дата обращения: 05.11.2024).

12. Давыденко М. С. Проверка на однородность экономических показателей // Новые математические методы и компьютерные технологии в проектировании, производстве и научных исследованиях. – 2016. – С. 19. URL: https://elib.gsu.by/bitstream/123456789/16973/1/Новые_математические_методы_Часть_2_2016.pdf#page=20 (дата обращения: 05.11.2024).

13. Зеневич А. О. и др. Редак ционн ая ко лл егия. URL: https://www.researchgate.net/profile/Shadiye-Sultanova/publication/375999318_sovremennie_sredstva_sviazi/links/6566ee54b1398a779dc3a1f7/sovremennie-sredstva-sviazi.pdf (дата обращения: 05.11.2024).

14. Израилов К. Е. МОДЕЛИРОВАНИЕ ПРОГРАММЫ С УЯЗВИМОСТЯМИ С ПОЗИЦИИ ЭВОЛЮЦИИ ЕЕ ПРЕДСТАВЛЕНИЙ. ЧАСТЬ 1. СХЕМА ЖИЗНЕННОГО ЦИКЛА // Труды учебных заведений связи. 2023. №1. URL: https://cyberleninka.ru/article/n/modelirovanie-programmy-s-uyazvimostyami-s-pozitsii-evolyutsii-ee-predstavleniy-chast-1-shema-zhiznennogo-tsikla (дата обращения: 05.11.2024).

15. Калевко Виктор Васильевич, Лагерев Дмитрий Григорьевич, Подвесовский Александр Георгиевич Управление образовательной программой вузов в контексте подготовки конкурентоспособных разработчиков программного обеспечения // Современные информационные технологии и ИТ-образование. 2018. №4. URL: https://cyberleninka.ru/article/n/upravlenie-obrazovatelnoy-programmoy-vuzov-v-kontekste-podgotovki-konkurentosposobnyh-razrabotchikov-programmnogo-obespecheniya (дата обращения: 05.11.2024).

16. Ключников Евгений Александрович Повышение производительности веб-приложения // Вестник Сыктывкарского университета. Серия 1. Математика. Механика. Информатика. 2010. №12. URL:

https://cyberleninka.ru/article/n/povyshenie-proizvoditelnosti-veb-prilozheniya (дата обращения: 05.11.2024).

17. Когут А. Е. Разработка системы «HR-automation». – 2023. URL: https://earchive.tpu.ru/handle/11683/76278 (дата обращения: 05.11.2024).