

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

Кафедра _____ «Прикладная математика и информатика» _____
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки / специальности)

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Система наполнения базы данных практик»

| | |
|--------------|--|
| Обучающийся | _____ А.Н. Ермолаев _____ (Инициалы Фамилия) (личная подпись) |
| Руководитель | _____ М.А. Тренина _____ (ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия) |
| Консультант | _____ к.п.н., доцент А.В. Егорова _____ (ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия) |

Тольятти 2024

Аннотация

Тема: «Система наполнения базы данных практик».

В данной выпускной квалификационной работе исследуется реализация системы наполнения базы данных практик. В исследовании рассматриваются задачи исследования требований к системе, разработка, сборка и тестирование системы наполнения базы данных практик.

Структура выпускной квалификационной работы представлена введением, тремя разделами, заключением, списком используемой литературы и листингом программного кода.

Во введении описывается актуальность проводимого исследования, заключающаяся в отсутствии подходящей, современной системы наполнения базы данных практик.

В первом разделе описывается анализ современных систем наполнения баз данных, рассматриваются их возможности и недостатки.

Во втором разделе описывается проектирование системы наполнения базы данных практик, выявляются функциональные требования к системе, указывается технологический стек.

В третьем разделе описывается процесс разработки frontend, процесс разработки backend, сборка и тестирования системы. Производится сравнение методов добавления данных в систему наполнения базы данных практик.

В заключении был сделан вывод, что созданная в результате исследования система наполнения базы данных практик полностью соответствует заявленным требованиям. Она способна упростить процесс добавления практик в базу данных, снизить количество затраченного пользователями времени на выполнение этого действия.

Данная бакалаврская работа включает в себя 57 страниц исследования, 22 рисунка, 5 таблиц, списка 25 источника, 1 приложения в виде листинга программы.

Abstract

Topic: "The system of filling the database of practices".

In this work, the implementation of the system of filling the database of practices is investigated. The research deals with the tasks of researching system requirements, developing, assembling and testing a system for filling a database of practices.

The structure of the work is represented by an introduction, three chapters, a conclusion, a list of used literature and a listing of the program code.

The introduction describes the relevance of the research, which consists in the absence of a suitable, modern system for filling the database of practices. The first chapter describes the analysis of modern database filling systems, discusses their capabilities and disadvantages.

The second chapter describes the design of the practice database filling system, identifies the functional requirements for the system, and specifies the technological stack.

The third chapter describes the frontend development process, the backend development process, the assembly and testing of the system. The methods of adding data to the system of filling the database of practices are compared.

In conclusion, it was concluded that the system of filling the database of practices created as a result of the study fully meets the stated requirements. It can simplify the process of adding practices to the database, reduce the amount of time spent by users to perform this action.

This work includes research 57 pages, 20 pictures, 5 tables, a list of 25 sources, 1 appendix in the form of a listing of the program.

Содержание

| | |
|--|----|
| Введение..... | 5 |
| 1. Анализ современных систем наполнения баз данных | 8 |
| 1.1. Анализ используемой системы и ее проблемы..... | 8 |
| 1.2. Анализ методов наполнения базы данных | 11 |
| 1.3. Представление собственной системы..... | 15 |
| 2. Проектирование системы наполнения базы данных практик | 17 |
| 2.1. Функциональные требования к разрабатываемой системе | 17 |
| 2.2. Требования к производительности системы..... | 23 |
| 2.3. Описание архитектуры системы и основных компонентов | 25 |
| 2.4. Технологический стек и инструменты разработки | 29 |
| 2.5. Последовательность разработки проекта | 34 |
| 3. Разработка, сборка и тестирование системы..... | 36 |
| 3.2. Разработка frontend | 38 |
| 3.3. Сборка проекта..... | 42 |
| 3.4. Тестирование системы наполнения | 43 |
| Заключение | 53 |
| Список используемой литературы и используемых источников..... | 55 |

Введение

Практическая система наполнения баз данных является важнейшим компонентом современного управления данными. В нашем быстроразвивающемся мире информационных технологий, в цифровую эпоху, предприятия и организации в значительной степени полагаются на базы данных для хранения, управления и обработки огромных объемов информации. Исключением не стал и Тольяттинский государственный университет, использующий в качестве системы наполнения базы данных практик корпоративную систему дистанционного обучения «Росдистант». Она имеет ряд особенностей и преимуществ, однако не лишена недостатков, которые способны повлиять на учебный процесс. Такими недостатками являются скорость работы системы, невозможность импорта практик из внешних источников, экспорт данных.

Эта работа проведена с целью разработки системы наполнения базы данных практик по заказу отдела разработки информационных систем ТГУ. Система будет разработана таким образом, чтобы облегчить ввод данных в базу данных, связанных с различными практиками. Система будет иметь понятную авторизацию и регистрацию, будет предоставлять доступ к информации о различных практиках и разграничивать возможности взаимодействия между системой и пользователем на основании роли пользователя, что повысит эффективность организации и сохранит данные в безопасности. В этом дипломе будут рассмотрены различные компоненты, необходимые для разработки системы, включая структуру базы данных, пользовательский интерфейс и методы добавления практик. Работа будет включать в себя анализ используемых в университете систем наполнения баз данных, их преимуществ и недостатков, а также способов их улучшения для удовлетворения потребностей современного управления данными.

Благодаря этой работе скорость наполнения базы данных практик информацией возрастет, появятся новые возможности для преподавателей,

такие как импорт данных из внешних файлов и экспорт данных. В конечном счете, эта разработка внесет вклад в область управления данными и предоставит отделу по разработке информационных систем ТГУ практическое решение: систему наполнения базы данных практик.

Все вышесказанное определило актуальность моей темы исследования – система наполнения базы данных практик.

Объект исследования: система наполнения базы данных практик.

Предмет исследования: разработка программного обеспечения для управления данными, связанными с практиками студентов. В частности, проектирование и разработка системы, обеспечивающей возможность создания, изменения и удаления данных о практиках, а также быстрый доступ к этим данным.

Целью бакалаврской работы является разработка системы наполнения базы данных практик.

Чтобы выполнить все цели, был поставлен целый ряд задач, состоящий из:

- изучить требования к системе;
- выявить основные и дополнительные требования к системе;
- изучить рынок поставляемых средств разработки;
- изучить использование клиент-серверной архитектуры;
- составить дорожную карту разработки проекта;
- выбрать методику разработки;
- составить план-график разработки и тестирования;
- разработать систему наполнения базы данных практик;
- собрать систему;
- протестировать основной функционал системы.

Основными источниками данных для создания работы являются: Java–документация, tutorial HTML, гайд CSS, документация Maven, информационные порталы с разработкой программного обеспечения и различные интернет-источники.

Поставленные цели и задачи исследования, определили ее структуру, которая состоит из введения, трех разделов и заключения.

Во введении обосновываются актуальность работы, ее цель, задачи предмет и объект исследования.

В первом разделе описывается анализ современных систем наполнения баз данных.

Во втором разделе описывается проектирование системы наполнения базы данных практик, указываются функциональные требования.

В третьем разделе описывается процесс разработки, сборки и тестирования системы.

В заключении представлены результаты, выводы и дальнейшее развитие проекта.

1 Анализ современных систем наполнения баз данных

1.1 Анализ используемой системы и ее проблемы

В современном мире высоких технологий, где различные организации стремятся к увеличению производительности труда, чтобы экономить время своих сотрудников путем внедрения различных автоматизированных информационных систем и направлять его на более важные задачи, имеется потребность в системе наполнения базы данных практик. Тольяттинский государственный университет не отстает от современных тенденций и пытается использовать все преимущества информационных технологий.

Система наполнения базы данных – это программное обеспечение (ПО), которое имеет встроенный функционал для работы с информацией и ее управлением. Система наполнения базы данных практик – это сложный в реализации программный продукт, однако его использование необходимо, поскольку это несет в себе следующие преимущества в использовании:

- стандартизация данных;
- легкость внесения изменений;
- безопасность;
- ограниченный доступ к информации.

Так, в университете, в качестве системы наполнения базы данных практик используется внешняя платформа Росдистант. Росдистант – это большая образовательная платформа, которая позволяет студентам получать образование, не посещая здания организаций, то есть предоставляет услуги дистанционного обучения.

Несмотря на то, что данная платформа нужна для предоставления возможности дистанционной учебы, она включает в себя и функции системы наполнения базы данных практик. Тем самым позволяет сотрудникам и студентам ТГУ экономить собственное время. Вид на страницу системы со стороны студента указан на рисунке 1.

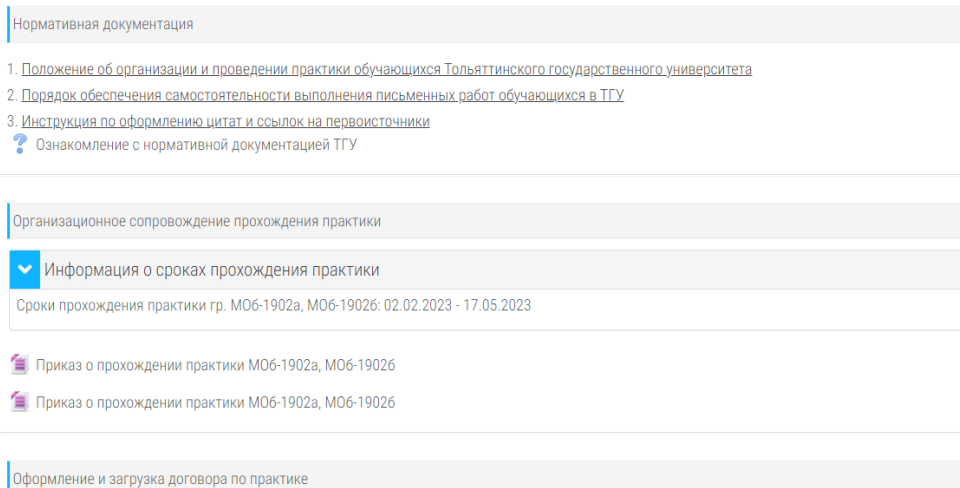


Рисунок 1 – Отображение практики в системе Росдистант

Выбранная университетом система, как и всякий программный продукт, обладает положительными свойствами. Перечислим положительные моменты платформы Росдистант, которые относятся к ее системе наполнения базы данных практик:

- заполнение информации о практике: преподаватель может заполнить информацию о прохождении практики, ее сроках начала и конца, названии и поставленных целей;
- оценка: система позволяет преподавателю оценивать работу студента и выставлять оценку;
- сбор практик: росдистант собирает практики в группы и хранит их;
- отчетность: существующее решение позволяет смотреть отчеты и создавать их, основываясь на имеющихся данных.

После перечисления вышеизложенных плюсов существующей системы наполнения базы данных практик можно было бы сделать вывод, что она идеально подходит под предъявляемые требования, однако это не так. Как и любое программное обеспечение, система наполнения базы данных практик платформы Росдистант имеет и негативные свойства. Перечислим их:

- неудовлетворительная скорость загрузки файла;
- медленный процесс создания практики;
- долгое ожидание при отправке практики студента на проверку;
- невозможность импортировать практики из внешних источников;
- интуитивно-непонятный интерфейс.

Теперь подробнее разберем каждый представленный недостаток. Начнем с неудовлетворительной скорости загрузки файла. Чтобы студенту отправить файл, нужно затратить больше времени, чем хотелось бы. Этот недостаток связан с неэффективными методами проверки соответствия файла.

Следующим описанным недостатком существующего решения является медленная отправка практики студента на проверку, а также долгое ожидание отправки практики преподавателю. Эти негативные черты системы тесно связаны с предыдущей проблемой и имеют общие корни, а именно неэффективные алгоритмы проверки вводимых данных, а также отсутствие проверки на стороне клиента, в среде frontend.

Используемая в университете система Росдистант не позволяет преподавателям и администраторам осуществлять импорт данных о практиках из внешних источников. Эта отрицательная черта ПО характеризуется масштабом платформы, и ее разработчики не успевают создавать улучшения этой части платформы, системы наполнения базы данных практик, поскольку ставят перед собой более важные задачи, охватывающие большую аудиторию.

Последним найденным недостатком используемой системы является интуитивно-непонятный интерфейс. Поскольку выбранная в качестве системы наполнения базы данных практик платформа Росдистант ставит в приоритет именно организацию дистанционного образования, а не постоянное развитие системы наполнения базы данных практик, то из-за ее масштабов не уделяется должного внимания визуальному оформлению системы. Это приводит к тому, что пользователи системы вынуждены

тратить большое количество рабочего времени на изучение функционала системы и адаптацию к ее непонятному интерфейсу.

1.2 Анализ методов наполнения базы данных

Система заполнения базы данных практики является важнейшим компонентом любой организации, желающей эффективно управлять своими данными. В современном быстро меняющемся мире, где объем генерируемых данных огромен, наличие надежной системы для заполнения базы данных практики имеет важное значение.

В наше время существует несколько подходов к наполнению базы данных практики. Перечислим их:

- ручной ввод данных;
- использование автоматизированных систем;
- импорт данных.

Рассмотрим каждый из вышеперечисленных методов более подробно, разберем основные положительные моменты при выборе метода, порядок действий пользователя при его использовании, укажем отрицательные качества выбранного метода. Первый подход для наполнения базы данных практик – это ручной ввод данных.

Ручное добавление данных – это процесс, при котором данные вводятся вручную сотрудниками в программу управления базой данных. Этот подход может быть полезен, когда необходимо добавить небольшое количество данных в базу данных, либо когда данные нужно быстро обновить или исправить.

Процесс ручного добавления данных начинается с открытия соответствующей формы или страницы для включения данных. Она может быть в виде интерфейса управления системы базами данных или может быть создана заранее и включать все необходимые поля заполнения. Например, для нашей темы, практики, форма может включать поля, такие как имя

студента, фамилия и отчество студента, его группу, описание задания практики, дата ее начала и конца, оценки за проделанную работу и группу студента. Пользователь должен заполнить все необходимые поля, введя данные. При этом соблюдаются требования к форматированию данных, чтобы избежать ошибок при работе с этими данными в будущем. После заполнения всех полей пользователь отправляет информацию для ее обработки в систему. Данные могут сохраняться разным множеством способов, например, путем нажатия кнопки "Сохранить" на форме.

Основные преимущества ручного добавления данных включают в себя возможность контроля пользователя за качеством вводимых данных, возможность быстрого обновления или исправления данных. Это позволяет пользователю иметь быстрый доступ к форматированию данных. Отлично подходит для добавления небольшого количества данных опытным, внимательным и ответственным сотрудником.

Помимо вышеперечисленных плюсов, данный метод также имеет ряд отрицательных моментов. Одним из ключевых отрицательных факторов является большое количество потраченного времени на добавление, изменение и удаление данных. Такие затраты времени далеко не всегда приемлемы и напрямую влияют на эффективность работы образовательной организации.

Следующим отрицательным фактором данного метода ручного ввода данных в базу данных практик является трудоемкость этого процесса. Сотрудник организации должен знать, какие форматы данных необходимо использовать, должен уметь разбираться в основах управления базами данных, знать предметную область.

Главным минусом ручного метода является возможность ввода ошибочных данных сотрудником. Это может затруднить работу организации, создать большое число вытекающего из этого проблем.

Второй подход заключается в использовании автоматизированных систем. Автоматический сбор данных может быть достигнут путем

использования различных инструментов и технологий. Автоматизированные системы используют программное обеспечение для ввода значений в базу данных, что сокращает время, необходимое для ввода данных, и устраняет ошибки. Эти системы часто используют шаблоны, чтобы гарантировать согласованность и точность введенных данных.

Проведя анализ метода автоматического заполнения базы данных, можно сделать вывод что основными положительными качествами являются скорость заполнения, устранение некоторых ошибок, возникающих при ручном вводе данных, проверка на соответствие формата данных. Эти преимущества позволяют избегать создания большого числа ошибок, предоставляют пользователю удобство в заполнении базы данных.

Кроме вышеперечисленных положительных качеств, автоматический способ заполнения базы данных практик имеет отрицательные черты. К одним из таких можно отнести качество полученных или введенных данных. Из-за ошибок в ПО, может возникнуть ситуация, когда программа собирала неверные данные, не соответствующие требуемому формату или данные были повреждены в ходе их обработки.

Следующим отрицательным качеством является неполнота данных. Автоматическая система может пропустить некоторые требуемые поля, вследствие чего поле может остаться пустым.

Помимо этого, на автоматические системы могут быть наложены правовые ограничения. Сбор данных может нарушать правовые нормы, если не соблюдаются правила конфиденциальности. Где-то потребуются ручное заполнение или дополнительный анализ данных.

Обновление данных. Автоматический сбор данных может быть недостаточным для периодического обновления базы данных. Однако, в зависимости от сложности системы и количества данных, может понадобиться много времени для обновления базы данных вручную.

Сложность реализации. Разработка автоматической системы сбора данных может быть очень трудоемкой задачей, особенно если это связано с

большим объемом источников данных. Требуется высокая квалификация разработчиков и значительные затраты на инфраструктуру и программное обеспечение.

Последний метод наполнения – импорт практик из файлов. Импорт из файлов для наполнения базы данных – это процесс загрузки информации из файлов различных форматов в нашу систему, а затем в составные части базы данных. Обычно импортируются файлы с форматами .XLS, .XLSX, .CSV. Этот процесс может быть реализован огромным количеством способов, на разных языках программирования, с использованием различных инструментов разработчиков. Чтобы импортировать данные из файла в базу данных практик, необходимо проделать следующие шаги:

- определить формат файла;
- сделать сортировку данных, описанную в требованиях;
- проверить корректность данных и их степень заполненности;
- загрузить файл в программу;
- получить желаемый результат.

Использование в работе метода импорта данных из файлов в базу данных дает свои положительные моменты. Теперь разберем их. Одним из положительных качеств выбранного способа является экономия времени и упрощение порога вхождения в программу сотрудников организации. Загрузка данных позволяет клиенту быстро загрузить большой объем информации в базу данных, без необходимости вводить данные вручную.

Третья хорошая особенность – это большая точность загружаемых данных. Если источник данных уже проверен и подготовлен к импорту, то клиент может быть уверен в том, что данные будут импортированы с максимальной точностью.

Четвертым потенциальным плюсом будет критерий использования. Импорт данных из файлов будет очень удобным способом загрузки данных, поскольку можно использовать различные форматы файлов (CSV, XLS, XML, JSON).

Последним плюсом этого метода является компактность. Использование этого метода позволяет сократить количество источников данных, необходимых для наполнения базы данных. Это может быть особенно полезно при работе с большими объемами информации.

Все вышеописанные методы заполнения базы данных имеют положительные и отрицательные качества своей работы. Это дает разработчику преимущество: возможность выбора. Для создания гибкой и удобной системы заполнения базы данных практик может пригодиться все. Благодаря этому анализу, можно создать собственную систему, включающую в себя сразу несколько методов заполнения базы данных. Выбор метода для заполнения базы данных практики зависит от следующего: размер организации, объема генерируемых данных, сложности данных и уровня квалификации персонала. Поэтому необходимо учитывать эти факторы перед выбором способов для заполнения базы данных практики.

1.3 Представление собственной системы

Проанализировав существующую систему, которую использует Тольяттинский государственный университет, было принято решение создать собственную систему, которая не будет иметь приведенных выше недостатков. Новая система будет учитывать требования бизнес-процессов этого университета. Это позволит создать продукт отвечающим всем требованиям заказчика.

Опишем главные методы внесения практик в базу данных. В разрабатываемой системе будут использоваться три метода наполнения данных: ручной ввод, автоматическое создание практики на группу и импорт практик из внешних файлов. Используя все методы сразу, позволим системе быть эффективной и нести в себе весь необходимый для этого функционал.

Также нужно искоренить еще один недостаток существующей системы – медленную скорость загрузки и обработки данных. Этого можно добиться,

используя встроенные функции языка, не подключая при этом лишние строки кода. Улучшение алгоритмов проверки поможет уменьшить затраченное на это время. Помимо вышеперечисленного, можно использовать различные средства проверки на стороне клиента, а не на стороне сервера. Это позволит значительно улучшить временные показатели нашего программного продукта.

Следующую проблему, которую решит наша система наполнения будет проблема недружественного интерфейса. Это один из основных отталкивающих факторов существующего решения, от которого новая система будет призвана избавиться. Для решения будем использовать простой дизайн, отсутствие нагромождений визуальных элементов, а также откажемся от реализации сложной анимации.

Нужно решить проблему импорта данных. Используя инструменты разработки, надо создать компонент новой системы, который будет способен импортировать информацию из загружаемых файлов, обрабатывать ее и добавлять в базу данных практик.

Система, описанная выше, будет иметь достаточный функционал. Благодаря такому набору решаемых проблем разрабатываемая система наполнения практик способна удовлетворить требования заказчика в ТГУ.

Выводы по первому разделу

Рассмотрев все аспекты, рассмотренные в данном разделе, можно с уверенностью утверждать, что используемая в Тольяттинском государственном университете система наполнения баз данных имеет свои преимущества, однако ее недостатки перевешивают в сторону разработки собственной системы наполнения базы данных практик. Именно эти факты позволяют заняться подготовкой и разработкой нового решения.

2 Проектирование системы наполнения базы данных практик

2.1 Функциональные требования к разрабатываемой системе

Разрабатываемая система наполнения базы данных практиками предназначена для повышения эффективности ввода и поиска данных. В этом разделе нами будет приведено описание требований к функциональным возможностям системы, необходимым для достижения ее корректной работы.

Функциональные требования являются самым важным компонентом любого проекта разработки, поскольку способны определить функционал программы. Они играют жизненно важную роль проекта в создании системы наполнения БД практики. Четко определенный функционал, как и тип системы, способен сделать так, чтобы помочь пользователям управлять данными более комфортно, повышать точность данных в различных условиях. Когда дело доходит до разработки системы наполнения БД практик, разработчикам необходимо учитывать несколько ключевых функциональных требований. Первым будет ввод и хранение данных. Система должна давать разным пользователям разные возможности. Так, преподаватели получают функции, нужные чтобы видеть данные студентов, создавать свои практики из различных источников и хранить в удобном для поиска виде. Для этого придется проектировать процесс создания схемы базы данных, достаточно гибкий. Он должен будет обеспечивать функциями для размещения различных типов данных и пользователей.

Следующим и очень важным функциональным требованием проекта будет сохранение безопасности введенных данных. Разрабатываемая мною система спроектирована так, чтобы с начального уровня защищать конфиденциальную информацию и уметь предотвращать несанкционированный доступ злоумышленников. Это означает, что разработчику можно включить в проект процессы внедрения системы

аутентификации пользователей и контроля доступа. Использование шифрования и протоколов безопасности позволит улучшить защиту данных на базовом уровне. Помимо безопасности, функциональные требования проекта, которые могут быть важны для разрабатываемой системы, будут включать масштабируемость, простоту использования и интеграцию с другими программными средствами. Систему нужно спроектировать так, чтобы она могла адаптироваться к росту числа пользователей, быть способной обрабатывать постоянно увеличивающиеся объемы поступающей информации. Помимо вышеперечисленного, интерфейс разрабатываемой программы должен иметь определенные свойства. Он обязан быть удобным, красивым и интуитивно-понятным. В интерфейсе пользователь должен чувствовать себя уверенно, легко ориентироваться и понимать основные принципы работы, чтобы упростить поток данных и уменьшить дублирование усилий. Кроме внешнего вида, наша работа должна быть готова к запуску на различных устройствах и операционных системах – должна быть многоплатформенной.

В целом, практическая система заполнения БД является важным инструментом для управления и организации данных в различных условиях. Она выполняет главные функциональные требования:

- ввод и хранение данных;
- безопасность;
- отчетность и анализ;
- масштабируемость;
- простота использования
- интеграция с другими системами;
- удобство пользователей.

Поскольку наша система, предназначена для предоставления централизованной платформы, то она должна позволять сотрудникам управлять, добавлять, удалять, вносить изменения. Это позволит

пользователям систематизировать информацию о практиках. На основании пройденных этапов проектирования, нами были составлены четкие требования к системе. Одними из таких требований и являются функциональные требования к проектируемой системе наполнения БД практик. Функциональные требования должны быть четко определены, чтобы проект был способен отвечать потребностям наших пользователей. Эти требования заключаются в следующем и изображены на рисунке 2.



Рисунок 2 – Модули функциональных требований

Перечислим список функций этих модулей:

- механизм управления аккаунтами;
- авторизация пользователей;
- ввод данных;
- импорт данных из внешних источников;
- валидация данных;
- безопасность данных;
- резервное восстановление;
- графический интерфейс.

Начнем разбор схемы с первого модуля. Модуль управления пользователями должен представлять из себя простой механизм, который помогает производить действия над учетной записью. Каждый клиент

должен быть авторизованным, т.е. должен иметь уникальный идентификационный набор текста для доступа к системе. Система должна позволять пользователям вводить собственные данные о практике. Доступные поля для заполнения: название, описание и тип практики. Нужен механизм для импорта записей из файлов пользователя. Согласно схеме, валидатор будет неотъемлемой частью проектируемой системы. Разработка должна верифицировать данные, введенные пользователями, уметь убеждаться, что они соответствуют требуемым стандартам. Безопасность данных является важным требованием в проекте. Обеспечение безопасности данных практик, представлено в виде механизма авторизации и шифрования информации. Необходима реализация строгого контроля доступа для предотвращения несанкционированного доступа к проекту и его данным. Для этого можем использовать систему распределения ролей между пользователями.

Кроме основных требований, можно также применить резервное восстановление. Программа, которую мы разрабатываем, будет иметь модуль копирования, который позволит создавать копии практических данных. Исключая простое копирование, будет предусмотрен механизм восстановления данных. Он будет использоваться в случае системного сбоя или потери данных. Понимание пользователя было причислено к этому списку не просто так.

Ведь разработка обязана иметь интуитивно понятный принцип работы, что и будет отличать систему от конкурентов. Также, это позволит избежать создания информационного модуля, выполняющего функции поддержки и подсказки выполнения действий.

Чтобы реализовать работу и поддержку вышеперечисленных возможностей, нужно перечислить функциональные требования. Они представлены в таблице 1.

Таблица 1 – Функциональные требования к системе

| Функциональное требование | Описание |
|--|---|
| Отождествление пользователя | Проектируемая система проверяет логин и пароль пользователя, перед доступом к каким-либо данным. |
| Управление разрешениями | Администраторы могут устанавливать роли и доступ к полям БД. |
| Ввод данных практики | Идентифицированные пользователи, имеющие к этому доступ, могут вводить новые данные практик. |
| Назначение даты начала и конца практики | Авторизованные пользователи способны сами указывать даты начала и конца практики. |
| Отображение практик | Авторизованные преподаватели имеют возможность просмотра, удаления практики студента. |
| Кроссплатформенность системы | Система наполнения доступна для использования на различных устройствах, таких как: персональный компьютер, смартфон и др. |
| Резервное копирование и безопасность полученных данных | Система обеспечивает резервное копирование наших практик и меры безопасности, включая, шифрование конфиденциальных данных и контроль доступа для предотвращения несанкционированного доступа или потери данных. |

Разрабатываемая программа может иметь функции, связанные с распределением ролей различных типов пользователей. Так, например пользователь с ролью «Преподаватель» должен иметь доступ к одним функциям, таким как: добавление своей практики, ее изменение, удаление, поиск и т.д. Более подробные функции роли указаны на рисунке 3.

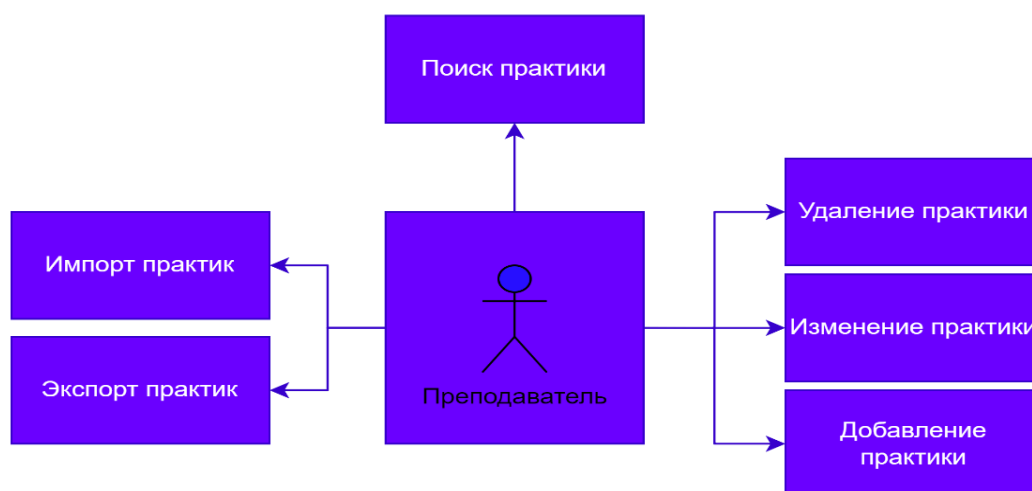


Рисунок 3 – Возможности роли преподавателя

Студент должен иметь функции просмотра только собственных практик. Администратор должен иметь помимо функций, схожих с возможностями преподавателя, следующие возможности: блокировка пользователей, удаление любой практики, просмотр любой практики, экспорт всей базы, загрузка БД. Более подробные функции распределения ролей указаны на рисунке 4 и рисунке 5.

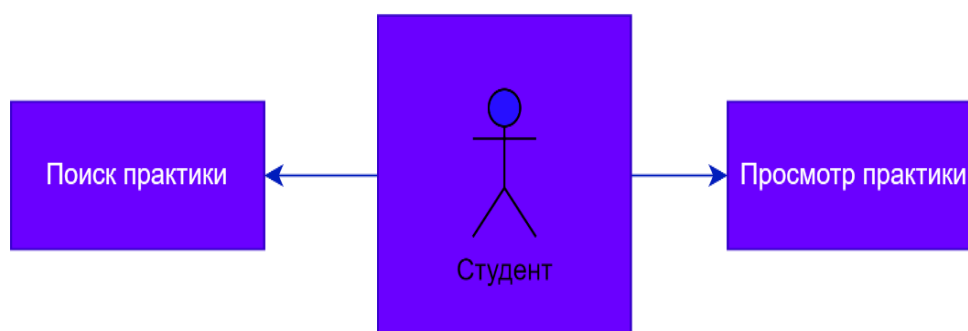


Рисунок 4 – Возможности роли студента

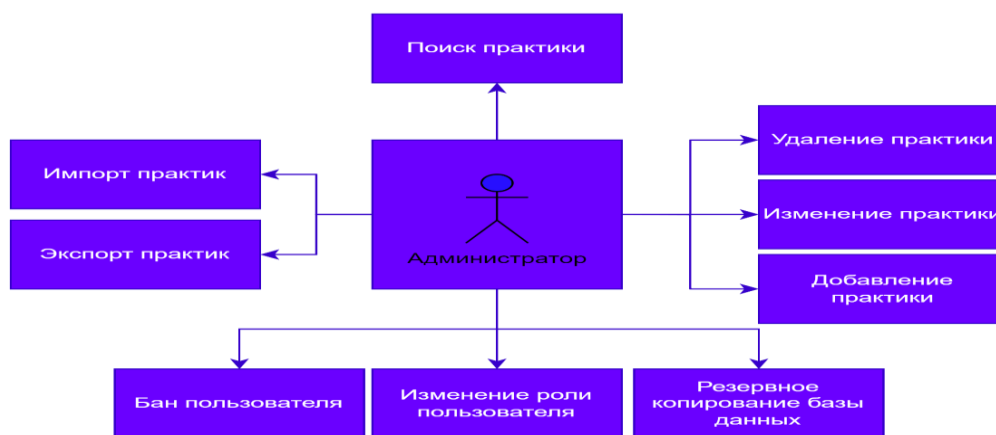


Рисунок 5 – Возможности роли администратора

В вышеизложенном разделе, были описаны характерные функциональные требования к системе наполнения базы данных практик. Именно такой список критериев достаточен и необходим для того чтобы обеспечить соответствие ожидания ее пользователей с реальностью. Эти

требования представляют собой ряд специфичных возможностей: управление пользователями, ввод практических данных, проверку данных в среде клиента, поиск и безопасность, резервное восстановление и отображение практик, а также красивый интерфейс для пользователей. Соответствуя этим требованиям, система обеспечит надежную и эффективную платформу для управления данными практики.

2.2 Требования к производительности системы

Требования к производительности важны для системы, основанной на взаимодействии с БД. Примером такой является система наполнения базы данных практик. Она предназначена для того, чтобы помочь пользователям вводить данные, которые затем можно применять для использования в различных целях, включая анализ, отчетность и принятие решений. Чтобы обеспечить оптимальную работу этой системы, необходимо установить требования к производительности, касающиеся различных аспектов. К таким относят скорость, надежность, масштабируемость и безопасность данных пользователей.

Первым и главным основным требованием к производительности системы заполнения базы данных практики является скорость. Пользователи ожидают, что система будет быстрой и отзывчивой, особенно при вводе относительно огромных объемов данных. Это означает, что система должна обрабатывать достаточно–необходимое количество одновременно–залогиненных пользователей без замедления или сбоя. Поэтому разрабатываемая система будет с использованием эффективных алгоритмов и оптимизированных запросов к базе данных, которые минимизируют время отклика. Кроме того, наш проект должен размещаться на надежной серверной инфраструктуре, способной обрабатывать быстрорастущие объемы трафика.

Масштабируемость для нашего проекта является важным требованием

к производительности системы наполнения базы данных практик. Так как рост пользователей будет происходить не постепенно, а волнообразно, то объем данных также увеличится, и система должна быть способна увеличивать или уменьшать свой масштаб в соответствии с потребностями. Это означает, что система предполагает в себе наличие учета масштабируемости с использованием таких технологий, как облачные вычисления и распределенные базы данных. Помимо этого, в разрабатываемом программном продукте будут присутствовать функции распределения нагрузки на сервера, позволяющие перенаправить трафик на другие вычислительные мощности.

Последней, но немаловажной является безопасность. Она является важнейшим требованием к производительности системы наполнения БД. Клиенты проекта должны быть уверены, что их конфиденциальные данные защищены от незаконного доступа, потери или кражи. Чтобы успешно выполнить это требование, система должна исполнять основные функции безопасности, такие как взаимодействие с брандмауэрами, шифрование, контроль доступа и механизмы аутентификации [14]. Кроме вышеперечисленных требований, система должна регулярно проверяться и тестироваться на наличие уязвимостей.

Определим требования к производительности системы:

- время ввода данных. Время, необходимое для ввода данных, необходимо свести к минимуму для повышения эффективности;
- время поиска и извлечения. Необходимо для поиска и извлечения данных, должно быть приравнено к минимуму для повышения производительности;
- обработка данных. Время, затраченное на обработку данных должно быть минимально. Программа должна оперативно и успешно влиять на данные, чтобы обеспечить своевременный доступ к информации;
- отклик пользователя. Система должна быстро реагировать на запросы пользователей, чтобы улучшить взаимодействие с

пользователем;

- постоянная работа. Система работает круглосуточно, чтобы обеспечить стабильно–непрерывный доступ к данным;
- масштабируемость. Разработка должна иметь функции расширения и уменьшения, то есть быть масштабируемой. Это позволит ей соответствовать будущему росту организации и ее требованиям к данным;
- копирование. Используя возможность резервного копирования, продукт гарантирует, что данные не будут потеряны в случае системного сбоя или аварии.

Требования к производительности являются главными с точки зрения описания функционала и имеют решающее значение для успеха проектируемой системы [15]. Обращаясь к таким показателям, как скорость, надежность, масштабируемость и безопасность, система может удовлетворить потребности пользователей и обеспечить надежный и эффективный метод ввода данных в базу данных. Очень важно проектировать систему с учетом этих требований, чтобы она могла обрабатывать нужные клиентам объемы данных и поддерживать множество пользователей, оставаясь при этом быстрой, надежной и безопасной.

2.3 Описание архитектуры системы и основных компонентов

Системная архитектура системы наполнения базы данных практики является значительной частью разрабатываемого проекта. Ее следует тщательно описать, чтобы обеспечить нам понимание ее функциональности. В этом разделе уделим внимание описанию архитектуры системы заполнения базы данных практики, которая предназначена для облегчения процесса сбора и управления данными.

Наша система будет состоять из списка нескольких компонентов, которые обеспечивают работу, взаимодействуя между собой для достижения

этой цели. Эти компоненты включают пользовательский интерфейс, систему управления базами данных, систему контроля и обеспечения безопасности [16]. Она использует путь внедрения уровня доступа к данным по ролям и сервер приложений. Пользовательский интерфейс будет важной визуальной частью проекта, ведь он использует графический компонент, который является прослойкой между пользователями и сервером, позволяя юзерам вводить и извлекать данные. Система управления базой данных отвечает за предоставления данных, их хранения, удаление и резервное копирование. Уровень доступа к данным предоставляет возможность изъять данные, находящиеся на хранении в БД системы, а сервер приложений отвечает за запуск приложения. Наш пользовательский интерфейс был спроектирован так, чтобы быть удобным для пользователей, а еще быть интуитивно понятным и отзывчивым. Поэтому он состоит из нескольких видимых пользователям форм, которые способны собирать данные. В проекте данные будут связаны с размещением информации студентов о практике, также имена, фамилии студентов, организация практики, имя руководителя и продолжительность практики. Формы разработаны так, чтобы в них было легко ориентироваться, поскольку в них встроены валидаторы, проверяющие введенные данные.

Система управления базой данных отвечает за возможности хранения и извлечение записей в виде текста и картинок. Данные будут связаны с местами практики студентов. В СУБД используется система самой распространенной реляционной базой данных, которая упрощает хранение, организацию и поиск данных. Логическая вариативность нашей БД спроектирована так, чтобы быть гибкой, позволяя добавлять и удалять поля записи по мере необходимости. Благодаря этому доступу к полям базы данных отвечают за предоставление доступа к данным, доступных для пользователя и хранящимся в гибкой базе данных. Он реализован с помощью набора технологий доступа и протоколов контроля подключения пользователей к БД. Использование совокупности мер предоставляют

функционал для пользователей: извлечение, выборка и хранение данных. В простых проектах для реализации этой возможности достаточно иметь соединитель. Эта часть приложения обеспечивает удобный для юзера механизм доступа к данным, гарантируя, что только авторизованные в системе пользователи могут просматривать и изменять данные. Идеальный вариант работы этого компонента приведен на рисунке 6.

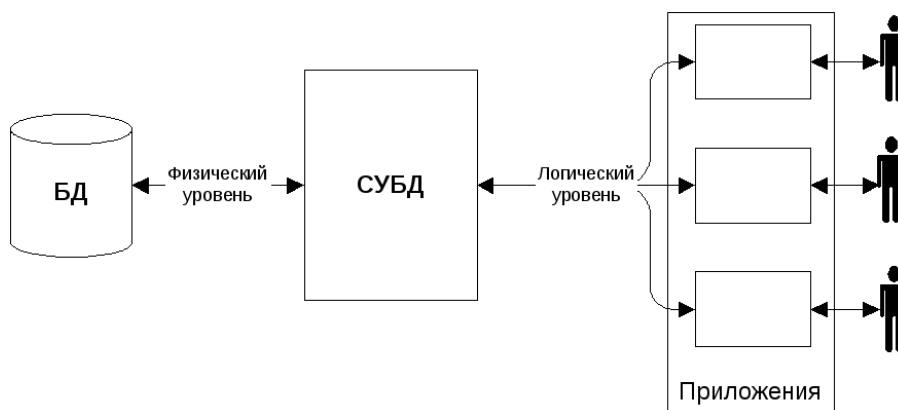


Рисунок 6 – Работа СУБД

Главным компонентом реализуемой системы будет сервер приложений. Он отвечает за размещение приложения в сети и дает ему ресурсы для работы. Сервер был настроен для запуска хостинга приложения так, что он взаимосвязан с СУБД. Компонент приложения способен обеспечивать нужную нам платформу для запуска основы, давая гарантию на защиту данных от несанкционированного доступа. Системная архитектура разрабатываемого проекта по практикам будет жизненно важной составляющей, которая облегчает сбор и управление данными. Обычно данные связаны с тематикой программы, поэтому это информация о студентах и их практиках. Система должна состоять из определенного разработчиком ряда компонентов. Эти компоненты должны работать вместе на постоянной основе, обеспечивая достижение целей. Включая пользовательский интерфейс, СУБД, возможности доступа к данным и

сервер системы. Архитектура спроектирована так, чтобы быть гибкой, безопасной и удобной для пользователя, что обеспечивает надежность и эффективность системы.

Поскольку система для заполнения базы данных практики представляет собой программное приложение, состоящее из нескольких основных компонентов, работающих вместе для достижения актуальной и точной информации. Тогда основными компонентами системы являются:

- пользовательский интерфейс;
- система управления базой данных;
- модуль ввода пользовательских данных;
- модуль редактирования данных;
- модуль представления.

Разбор начнем с пользовательского интерфейса. Он реализует графический интерфейс, понятный пользователям. Добивается этого благодаря целому ряду инструментов и языков программирования, после чего позволяет клиентам получать доступ к взаимодействию с функциями системы. Интерфейс включает в себя форму отображения визуального контента, картинки, таблицы, схемы, текстовые данные, меню используемое для ввода, редактирования и просмотра данных. Система управления БД, в отличие от интерфейса, дает плюсы и разработчику. Она ускоряет скорость разработки нашего проекта, ведь перекладывает часть функций на готовую систему. Этот компонент отвечает за управление базой данных практик, используя функции хранения, поиска и обновления данных. Использование его функций гарантирует нашей БД верно организованные механизмы запросов, доступа для авторизованных пользователей разрабатываемого проекта.

Модуль представления данных позволяет вводить новые, собственно—ручно написанные данные в базу данных практики с клиентской части выбранной клиент-серверной архитектуры. Он включает в себя такие разнообразные функции, как: проверка вводимых данных, проверка ошибок

и нормализация данных, гарантия корректности, согласованности введенных данных. Помимо этого, будем использовать и модуль редактирования данных. Создаваемый этап приложения позволяет авторизованным пользователям видоизменять существующие данные. Он определяет в нашем проекте наличие следующей важной функции: возможности контроля доступа к данным с клиентской стороны. Модуль способен обеспечивать гарантию того, что исключительно авторизованные пользователи могут вносить изменения в базу данных. Модуль представления представляет собой часть проекта, которая будет отвечать за визуальную часть пользовательских данных. Он способен выводить записи об аккаунтах.

Можно сказать, что создаваемая система заполнения базы данных практиками представляет из себя довольно сложное программное приложение. Для эффективной работы приложения потребуется интеграция сразу нескольких компонентов. Компоненты взаимодействуют друг с другом, с внешними данными и обеспечивают наполнение базы данных практики проверенной информацией, которую можно использовать для повышения качества образования и обучения студентов.

2.4 Технологический стек и инструменты разработки

Для того чтобы выполнить поставленные выше цели, нужно определиться с выбором технологического стека. Он должен быть выбран с учетом требований к проекту и количеству функций, необходимых для постоянной работы системы. Чтобы достичь этой цели использовались различные методы подбора ПО и его анализ. Разбор начнем с определения выбранного типа архитектуры.

Клиент-серверная архитектура – это самая популярная в интернете архитектура, предназначенная для построения веб-приложений. Она использует взаимодействие между собой двух частей: клиента и сервера. Клиент отправляет необходимые ему для выполнения поставленных задач

запросы на сервер, является пользователем. Он должен использовать собственную вычислительную машину, реже используется терминал. Сервер будет вторым компонентом. Это самая важная составляющая часть выбранного типа архитектуры. Сервер представляет собой электронную вычислительную машину огромной мощности, несопоставимой с домашними системами. Он выполняет все необходимые вычисления на своей стороне. Потом предоставляет данные обратно клиенту. Взаимодействие этих компонентов происходит с использованием HTTP-запросов, поскольку они имеют подходящие для этого функции.

После выбора архитектуры нам нужно сделать выбор системы взаимодействия между собой различных компонентов системы. Из множества известных претендентов была выбрана технология MVC. Это технология, которая представляет собой архитектуру взаимодействия компонентов между собой, используя ее особенности. Она расшифровывается как модель, вид, контроллер. Основное преимущество системы состоит в том, что она является гибкой и в отличие от своих монолитных конкурентов легко изменяема. MVC система состоит из следующих компонентов:

- модель;
- вид;
- контроллер.

Модель является составной частью архитектуры. Реализует представление таблиц реляционной базы данных. Для правильного использования нужно использовать аннотации. Вид также часть модели, он отвечает за визуальное представление данных системы. Позволяет пользователю физически управлять представлением, интерфейсом. Помимо этого, вид формирует и отправляет запросы следующему компоненту системы. Этим компонентом является контроллер. Контроллер будет последним составным компонентом выбранной архитектуры. Является самым важным компонентом, ведь выполняет функции предоставления пользователям данные. Помимо этого, контроллер проверяет данные и

определяет их дальнейшую судьбу – отправить данные пользователю или компоненту.

Основной частью выбора технологического стека будет выбор языка программирования [6]. Обычно он представляет собой строго определенный синтаксис и определяет правила построения методов и функций в коде. Разрабатываемое программное приложение было разработано с использованием Java. Потому что он популярен среди программистов. Язык объектно-ориентирован, широко используется для разработки сложных приложений. Помимо этого, Java был выбран из-за простоты использования, независимости от платформы пользователя и большого сообщества разработчиков.

Java – это не единственная часть проекта. Помимо языка программирования Java использовался фреймворк Spring [10]-[11]. Spring – уникальный в мире разработке фреймворк, который позволяет создавать приложения различных типов и назначений. Он имеет открытый исходный код, благодаря чему его можно оптимизировать под свои нужды. Выбор был сделан в пользу него из-за универсальности, его возможностей. Без дополнительных пакетов и расширений уже способен на многое, имеет постоянную поддержку разработчиками.

Система управления базами данных будет выбрана исходя из-за простоты использования. Ее нужно было выбирать, чтобы обеспечить ряд возможностей из коробки [8]. Поскольку данные приложения хранятся в системе управления реляционными базами данных, то лучше всего нам подходит MySQL. Это популярная СУБД, легкая в использовании и имеющая огромное сообщество пользователей. Она была выбрана в качестве основной системы управления базой данных за счет своей простоты, надежности и возможности писать SQL-запросы на диалекте MySQL.

ORM (объектно-реляционное отображение) [3] способно дать разработчику функционал для представления данных из БД в виде объектов [5].

Чтобы облегчить написание запросов, был выбран фреймворк Hibernate. Он позволяет программисту упрощать запросы. Используется для языка программирования Java, отлично сочетается с фреймворком Spring. Поможет нам не создавать собственную систему SQL-запросов, а использовать интуитивно–понятное представление в виде объектов. Выбор технологии позволит увеличить производительность системы, ведь Hibernate оптимизирует запросы перед отправкой их в систему управления базами данных.

Интегрированная среда разработки [4]. IDE была выбрана с учетом возможностей, доступных для разработчиков из коробки. Рассматривались различные варианты ПО, но выбор был сделан. Приложение было разработано с использованием IntelliJ IDEA. Это среда IDE, которая предоставляет огромный набор инструментов для взаимодействия и разработки на языке Java. Плюсом в ее пользу является поддержка языка Java из коробки, широкий круг функциональных возможностей. А именно: подключение к базе данных из среды разработки, возможность точно настроить проект, подключение к системе контроля версий Git-Hub. Не частым явлением будет магазин с широким списком устанавливаемых дополнительных плагинов. С помощью этой среды разработки мы сможем эффективно писать, тестировать и отлаживать написанный код.

Для разрабатываемого приложения будет нужна система контроля версий. Исходный код приложения должен управляться с помощью популярной и широко используемой системы контроля версий Git. Он позволяет разработчикам совместно работать над кодом, отслеживать изменения и управлять версиями кода [1]. У данной системы есть пара минусов, это отсутствие русского языка.

Немаловажным компонентом проекта будет пользовательский интерфейс (UI). Согласно требованиям проекта, UI должен быть простым, чтобы не нагружать работу системы. Будет удобным и информативно–понятным для среднестатистического пользователя. Чтобы обеспечить эти

качества, были выбраны следующие технологии: HTML, CSS, FreeMarker. HTML – это язык гипертекстовой разметки страницы, который будет необходим для создания основных страниц системы. В его основе лежит система тегов, именно от них зависит тип отображаемых данных. Чтобы интерфейс был приятен глазу, нужно усовершенствовать его внешний вид [2]. Для этого и будет использован язык стилей CSS. Он представляет собой набор списков, которые способны описывать различные визуальные изменения путем прикрепления к отдельным тегам языка гипертекстовой разметки. Может прикрепляться по названию, классу и идентификатору. С помощью этого инструмента будем создавать не только статичные объекты, но и анимированные, в том числе анимированные переходы.

Подходя к концу выбора стека, нужно определиться с последним инструментом. Это будет фреймворк FreeMarker. Он используется для визуального отображения данных. Фреймворк, использует различные шаблоны, схожие с XML. Позволяет вставлять в страницу языка гипертекстовой разметки не только статичные данные, но и динамические. Добивается FreeMarker этого благодаря встроенным специальным возможностям в виде различных команд и специальных символов. Благодаря разнообразному использованию технологий, разработчику будет легко и быстро создать приятный и удобный и понятный интерфейс [3].

Кроме выбора самого стека, разработчикам важен выбор метода разработки программного обеспечения. Нами была выбрана методология жизненного цикла разработки программного обеспечения. Приложение было разработано с использованием методологии Agile. Она делает упор на совместную работу команды, гибкость и итеративную разработку. Является популярной методологией, широко применяется в как небольших организациях, так и в крупных технологических компаниях.

В итоге, в исследуемой работе «Система наполнения баз данных практик» был использован хороший стек и широкий спектр технологий, инструментов для разработки надежного и эффективного программного

приложения. От языков программирования и IDE до систем управления базами данных и ORM – каждый инструмент играл решающую роль в процессе разработки [22]. Именно такой выбор технологического стека и инструментов разработки позволит нам разработать надежную, стабильную, производительную и безопасную систему практик, отвечающую заявленным требованиям [23].

2.5 Последовательность разработки проекта

Чтобы использовать выделенное время наиболее эффективно и не потратить его в пустую, нужно составить план выполнения задач, которого необходимо будет придерживаться в ходе разработки системы наполнения базы данных практик. Описание процесса реализации проекта будет включать определенные ниже шаги:

- обоснование требований и функциональных потребностей системы, которые могут включать типы данных;
- выбор определенной СУБД на основе требований к системе;
- разработка структуры и схемы базы данных, включая таблицы, поля и связи между ними;
- разработка специальных элементов системы, которая позволит осуществлять импорт данных из различных источников в базу данных;
- разработка системы экспорта данных, которая может экспортировать данные из БД на персональный компьютер администратора;
- разработка пользовательского интерфейса для удобного доступа и управления данными;
- тестирование системы на соответствие различным утвержденным требованиям и функциональным возможностям, а также ее производительности и безопасности. Этот этап может включать как

модульное тестирование системы, так и тестирование системы в целом.

Благодаря следованию вышеизложенным этапам, шаг за шагом можно осуществлять задуманное и создать качественную и удобную для использования систему наполнения базы данных практик. Ключевым моментом в этом процессе является понимание основ разработки и использование знаний, полученных в ходе обучения в образовательной организации. Необходимо использовать весь накопившийся опыт и создать достойную систему.

Выводы по второму разделу

В приведенном выше разделе мы описали основные требования к системе. Изучили как функциональные, так и требования по производительности, надежности и безопасности проектируемой системы наполнения.

Были изложены этапы процесса разработки системы, включающие в себя разработку функционала системы, ее внешнего вида, архитектуры, базы данных. Немаловажным пунктом в этом разделе стал и выбор технологического стека и инструментов разработки. Благодаря такой совокупности и разнообразия инструментов, у разработчика есть все шансы создать достойный и отвечающий всем требованиям качества программный продукт. Особое отношение необходимо будет выделить для тестирования системы и реализации планов ее дальнейшего сопровождения и обеспечения.

3 Разработка, сборка и тестирование системы

3.1 Разработка backend

На основании выбранной архитектуры веб-приложения, построенной на взаимодействии трех компонентов MVC – Model and View and Controller, написанных на Spring, разработчику в ходе реализации проекта необходимо использовать модульное разделение классов. Это разделение способствует усложнению процесса разработки, однако программисты, имеющие опыт работы с такими системами, не столкнутся с большими проблемами и быстро поймут логику работы. Такое разделение имеет большой плюс, превосходящий все недостатки: модульность. Благодаря разбиению логики на три части, вид, контроллер и модель, дальнейшее внесение изменений и обновление системы потребуют минимальное количество редактирования кода. Ниже приведен рисунок 7, на котором изображен пример кода компонента модели. Это помогает программисту создавать модель представления сущностей и таблиц из базы данных непосредственно в коде программы.

```
5 usages
@Entity
@Table(name = "groups")
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Groups {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id")
    private Long id;
    @Column(name = "name_group")
    private String name_group;
}
```

Рисунок 7 – Код модели «Группа»

Чтобы модель имела ряд функциональных возможностей и могла выполнять их, необходимо указать ей контроллер [9]. Контроллер способен обеспечивать взаимодействие между другими компонентами и хранит в себе вычислительные функции. Пример использования контроллера указан на рисунке 8.

```
@PostMapping("/registration")
public String createUser(Student student, Model model) {
    if (!userService.createStudent(student)) {
        model.addAttribute(
            attributeName: "errorMessage",
            attributeValue: "Пользователь с email: "
                + student.getEmail() +
                " уже существует");
        return "registration";
    }
    return "redirect:/login";
}
```

Рисунок 8 – Регистрация студента

Для удобства разработки, функции, отвечающие за обработку данных, были вынесены в другой класс, который имеет в своем названии аббревиатуру «Service». Именно в таких классах происходит основная работа backend. Так, например, на рисунке 9 изображена функция изменения роли преподавателя, которая вызывается контроллером, принимающим метод POST из компонента вида.

```
public void changeStudentRoles(Student user, Map<String, String> form) {
    Set<String> roles = Arrays.stream(Role.values()) Stream<Role>
        .map(Role::name) Stream<String>
        .collect(Collectors.toSet());
    user.getRoles().clear();
    for (String key : form.keySet()) {
        if (roles.contains(key)) {
            user.getRoles().add(Role.valueOf(key));
        }
    }
    studentRepository.save(user);
}
```

Рисунок 9 – Функция класса Service

Благодаря такому взаимодействию различных классов, которые описывают три компонента MVC, удалось написать backend-часть системы наполнения базы данных практик. Используя знания и опыт, полученные за годы обучения, разработчик способен создать продукт, отвечающий всем заявленным требованиям и реализовать следующие функции: авторизацию, регистрацию, создание практики для студента и группы, импорт практик в базу данных, а также список различных функций администратора, включающие блокировку пользователей и изменение их ролей и множества других функций.

3.2 Разработка frontend

Чтобы пользователи разрабатываемого программного продукта могли без обучения использовать все его возможности, необходимо описать пользовательский интерфейс [19]. Согласно заявленным требованиям, он должен быть удобным, визуально приятным и интуитивно понятным для пользователей, имеющих средний опыт взаимодействия с веб-приложениями. Чтобы эти требования можно было выполнить, были использованы следующие технологии и инструменты:

- HTML – язык разметки, призванный описать основу страницы, ее каркас [17]-[18];
- CSS – язык стиля, используется для придания странице эстетически-приятного вида [20]-[21];
- FreeMarker – шаблонизатор, позволяющий отображать динамические данные на странице HTML.

Разработка страницы начинается с создания HTML-страницы. Для того чтобы страница поддерживала все доступные возможности, необходимо указать теги <HTML>, который указывает, что все его содержимое относится к одноименному языку. Тег <head> будем использовать для отображения и подключения метаданных, которые не всегда будут видны пользователям, за

некоторым исключением. Тег `<title>` является таким, позволяя пользователю видеть название странице, отображая его в виде надписи вкладки браузера. В `<body>` пишутся все остальные элементы, которые необходимы для визуального отображения. Пример разработанной веб-страницы, использующей HTML разметку, указан на рисунке 10.

```
<!DOCTYPE html>
<html>
<head>
  <title>Ermola</title>
</head>
<body>

<h4 style="color: red">Администратор еще не проверил ваши данные. После проверки вы получите доступ к аккаунту</h4>

</body>
</html>
```

Рисунок 10 – HTML-страница проекта

Следующим шагом в нашей frontend разработке будет визуальное преобразование веб-страницы. Используя CSS, язык стилей, можно добиться приемлемого результата, который будет соответствовать требованиям проекта. Язык стилей представляет из себя набор параметров и характеристик, которые применяются к тегам HTML, ссылаясь на них в различных видах [12]. Применение возможно не только по названию элемента, его классу, идентификатору, но и ко всему содержимому. Чтобы применить ряд параметров визуального стиля ко всей странице сразу, необходимо использовать специальный символ-обращение «*». Пример такого использования представлен на рисунке 11.

```
*, *:before, *:after {
  box-sizing: border-box;

  margin: 0;
  font-family: 'Abel', sans-serif;;

  font-size: 15px;
  line-height: 1.6;

  color: #333;
}
```

Рисунок 11 – Стиль ко всему содержимому

Помимо придания различных параметров тегу, с помощью CSS можно написать анимации. Для этого понадобятся средние навыки программирования и понимания логики языка стилей. Приведем пример, фрагмент кода, позволяющий создать анимацию, указанную на рисунке 12.

```
.oblast_HrefToGlav
{
  width: 100%;
  height: 100%;
}

.HrefToGlav
{
  text-align: center;
  width: 25%;
  height: 25px;
  color: black;
  font-size: 26px;
  margin-top: 50px;
  margin-left: 50px;
  text-decoration: none;

  position: relative;
  font-size: 20px;
  line-height: 20px;
  padding: 12px 30px;
  /* color: #FFF; */
  font-weight: bold;
  text-transform: uppercase;

  font-family: 'Roboto', Tahoma, sans-serif;
  background: white;
  cursor: pointer;
  border: 2px solid black;
}

.HrefToGlav:hover,
.HrefToGlav:active,
```



```

.HrefToGlav:focus
{
    color: black;
}

.HrefToGlav:after,
.HrefToGlav:before
{
    position: absolute;
    height: 4px;
    left: 50%;
    background: black;
    bottom: -6px;
    content: "";

    transition: all 280ms ease-in-out;
    width: 0;
}

```



Рисунок 12 – Анимация на основе CSS

Применение этих простых и универсальных инструментов позволяет создать удобную веб-страницу, имеющую приличный дизайн. Однако использование HTML и CSS не является универсальным решением всех проблем, связанных с веб-разработкой. Ведь помимо статичных данных, разрабатываемая система наполнения должна иметь возможность предоставлять пользователю набор динамических данных. Для этого мы используем FreeMarker. FreeMarker использует различные шаблоны, специальные символы, логические функции для отображения динамических данных [13]. Так, например, для отображения фамилии, имени и отчества авторизованного пользователя, а также его роли, мы используем взаимодействие логических функций шаблонизатора и его специальных

выражений. Результат приведенного ниже кода изображен на рисунке 13. А теперь приведем пример использования:

```
<div class="FIO">
  <#if student.zvanie=="Студент">
    ${student.zvanie}:
    <a>${student.familia}                ${student.name}
    ${student.otchestvo}</a>
  </#if>

  <#if teacher.zvanie=="Преподаватель">
    ${teacher.zvanie}:
    <a>${teacher.familia}                ${teacher.name}
    ${teacher.otchestvo}</a>
  </#if>
</div>
```

ПРЕПОДАВАТЕЛЬ:

Преподаватель: Сидоров
Владимир Кедрович

СТУДЕНТ:

Студент: Ильин Керам
Витальевич

Рисунок 13 – Динамическое отображение данных

Благодаря совокупности использования этих трех инструментов и технологий, мы создали пользовательский интерфейс, отвечающих всем заявленным первоначальным требованиям проекта. Взаимодействие технологий позволяет проектировать и разрабатывать визуально-приятное оформление интерфейса, проверять роли пользователей и делать многое другое.

3.3 Сборка проекта

В ходе разработки проекта не стоит забывать об этапе сборки. В стандартных и маленьких по размеру проектах, использующих ограниченное количество зависимостей, классов и фреймворков, нужно запустить главный класс, который имеет точку входа в проект. Однако, это не наш случай. Для

удобной, быстрой, а главное качественной сборки нужно иметь возможность автоматической сборки проекта. Можно написать собственный инструмент для сборки проекта, однако это трудозатратно и упущение некоторых перечислений или особенностей проекта может привести к фатальным ошибкам и в следствии неработоспособности системы.

Исходя из этого, в качестве сборщика проекта мы используем Maven [7]. Это фреймворк, которому разработчик делегирует функции компиляции, сборки, очистки памяти и удаления неиспользуемых зависимостей проекта.

Maven в автоматическом режиме сделает свою работу и выдаст состояние проекта в терминале нашей IDE. Пример вывода состояния проекта указан на рисунке 14.

```
2023-06-08 14:35:18.185 INFO 6788 --- [main] o.s.s.web.DefaultSecurityFilterChain : Will secure any request with [org.springframework.security.web.context
2023-06-08 14:35:18.425 INFO 6788 --- [main] o.s.h.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2023-06-08 14:35:18.634 INFO 6788 --- [main] com.example.buyseel.BuyseelApplication : Started BuyseelApplication in 4.569 seconds (JVM running for 5.287)
2023-06-08 14:35:32.274 INFO 6788 --- [nio-8080-exec-1] o.a.s.c.o.(Tomcat).(localhost).(/) : Initializing Spring DispatcherServlet 'dispatcherServlet'
2023-06-08 14:35:32.274 INFO 6788 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2023-06-08 14:35:32.275 INFO 6788 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
```

Рисунок 14 – Информация о запуске проекта от Maven

Таким образом, использование автоматического сборщика оправдывает себя. Позволяет в разы уменьшать время разработки проекта за счет автоматической сборки и дает возможность разработчикам сосредоточиться на других возможностях, которые требуют внимания.

3.4 Тестирование системы наполнения

Тестирование – это проверка на соответствие программного продукта с эталонным значением. Тестирование является неотъемлемой и очень важной составляющей разработки и может происходить как во время разработки, так и после нее. Главной целью тестирования является проверка продукта на

соответствие заявленным требованиям, обеспечивает проверку качества программного продукта. Оно включает в себя следующие виды: ручное и автоматизированное. Существует 7 принципов тестирования:

- стопроцентное тестирование невозможно;
- тестирование показывает наличие ошибок;
- ранее тестирование приветствуется;
- имеется парадокс пестицида;
- зависит от типа программы;
- накопление дефектов;
- ошибки есть всегда.

Тестирование проводилось на персональном компьютере разработчика, который имеет следующие системные характеристики, представленные в таблице 2.

Таблица 2 – Характеристики персонального компьютера

| Параметр | Компонент |
|----------------------|------------------------------|
| Операционная система | Windows 10, Домашняя |
| Процессор | Intel Core i5-10300H, 2.5GHz |
| Число ядер | 4+4 |
| Видеокарта | NVIDIA 1650TI, 4Gb |
| ОЗУ | 32Gb, DDR4 |
| Wi-Fi адаптер | 6.0V |
| Интернет-соединение | 100 Мбит/с |

Тестирование начнем с проверки авторизации. При вводе электронной почты, не укажем символ «@». Введем пароль и нажмем кнопку «Войти». Система проведет проверку на стороне клиента и выведет надпись о некорректном формате данных. Пример показан на рисунке 15.

Если же ввести корректное значение «123@mail.ru», то в строфе «Логин» появится зеленая галочка. Она является идентификатором правильно введенного значения. Пример этого указан на рисунке 16. Теперь, если нажать кнопку «Войти», то при корректном пароле произойдет

переадресация на главную страницу системы. Идентичная система проверки вводимых значений на стороне клиента применяется не только на странице авторизации, но и на остальных страницах разработанной системы наполнения данных практик, что делает демонстрацию этой функции на других страницах не нужной.

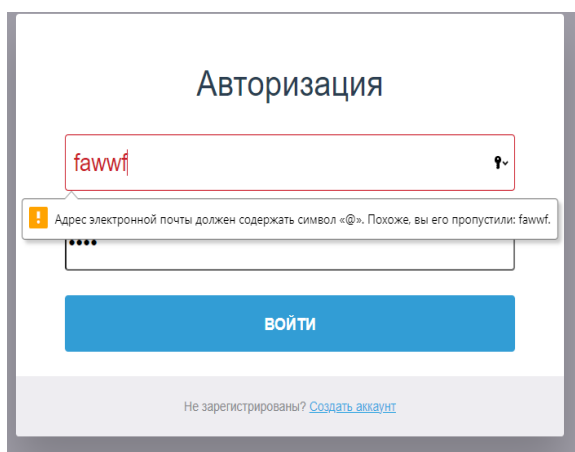


Рисунок 15 – Валидация на стороне клиента

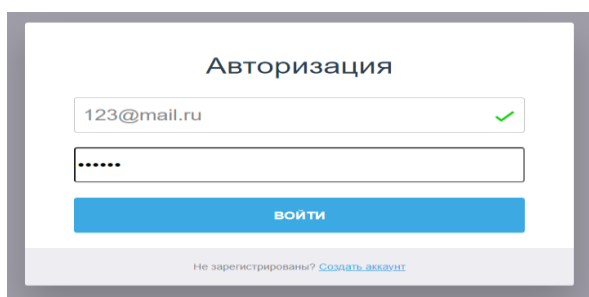


Рисунок 16 – Валидация при правильном значении

Теперь перейдем к тестированию главных функций нашего проекта, а именно добавлению практик в систему наполнения базы данных практик. Такое тестирование покажет нам на сколько наша система производительна и эффективна. Чтобы засечь время, затраченное на добавление данных, будем использовать функцию `Java System.nanoTime()`.

Тестирование будет проведено в несколько этапов: тестирование добавления практики студенту индивидуально, добавление практики группе студентов, импорт практик из внешних источников, сравнение эффективности методов заполнения базы данных практик. Было проведено первое тестирование и замер скорости работы функции, которое включает измерения добавление индивидуальной практики одному ученику. Все вводимые данные были одинаковы. Это сделано для чистоты эксперимента. Эти данные представлены на рисунке 17. В подтверждение проделанных действий, на рисунке 18 изображены поля базы данных системы практик.

Создание практики

Название:
Роман

Начало практики:
08.06.2023

Конец практики:
08.06.2023

Вид практики:
Семинар

Студент:
Ильин Керам Витальевич (888888)

Описание:
Прodelать работу...

СОЗДАТЬ

Рисунок 17 – Вводимые данные

| id | date_of_created | finish | name | oce | opisanie | sdi | start | vid | stuc | tea |
|-------|----------------------------|------------|-------|-----|---------------------|-----|------------|---------|------|-----|
| 11806 | 2023-06-08 21:26:43.817986 | 2023-06-08 | Роман | 0 | Прodelать работу... | 0 | 2023-06-09 | Семинар | 3 | 1 |
| 11807 | 2023-06-08 21:26:43.824967 | 2023-06-08 | Роман | 0 | Прodelать работу... | 0 | 2023-06-09 | Семинар | 4 | 1 |
| 11808 | 2023-06-08 21:26:43.830951 | 2023-06-08 | Роман | 0 | Прodelать работу... | 0 | 2023-06-09 | Семинар | 5 | 1 |
| 11809 | 2023-06-08 21:26:43.835938 | 2023-06-08 | Роман | 0 | Прodelать работу... | 0 | 2023-06-09 | Семинар | 7 | 1 |
| 11810 | 2023-06-08 21:26:43.841922 | 2023-06-08 | Роман | 0 | Прodelать работу... | 0 | 2023-06-09 | Семинар | 8 | 1 |
| 11811 | 2023-06-08 21:26:43.848943 | 2023-06-08 | Роман | 0 | Прodelать работу... | 0 | 2023-06-09 | Семинар | 9 | 1 |
| 11812 | 2023-06-08 21:26:43.854929 | 2023-06-08 | Роман | 0 | Прodelать работу... | 0 | 2023-06-09 | Семинар | 10 | 1 |
| 11813 | 2023-06-08 21:26:43.861768 | 2023-06-08 | Роман | 0 | Прodelать работу... | 0 | 2023-06-09 | Семинар | 11 | 1 |
| 11814 | 2023-06-08 21:26:43.867753 | 2023-06-08 | Роман | 0 | Прodelать работу... | 0 | 2023-06-09 | Семинар | 12 | 1 |
| 11815 | 2023-06-08 21:26:43.873927 | 2023-06-08 | Роман | 0 | Прodelать работу... | 0 | 2023-06-09 | Семинар | 13 | 1 |
| 11816 | 2023-06-08 21:27:18.963164 | 2023-06-08 | Керан | 0 | Прodelать работу... | 0 | 2023-06-08 | Семинар | 3 | 1 |

Рисунок 18 – Введенные в БД данные

Данная процедура добавления практики проводилась 10 раз. Ниже представлены промежуточные результаты, отображающие в себе то, с какой скоростью реализованная система способна обработать, принять и внести данные в базу данных. Замер времени добавления представлен на рисунок 19.



```
Добавление заняло: 7684800
```

Рисунок 19 – Пример результата в наносекундах

Промежуточные результаты замеров: 29,81 мс., 8,03 мс., 9,89 мс., 7,28 мс., 6,64 мс., 7,17 мс., 7,54 мс., 8,61 мс., 7,71 мс., 6,35 мс.

Исследуем приведенные данные: все запуски, в среднем, имеют примерно равную скорость добавления, однако, выделяется первое добавление. Оно имеет гораздо большее затраченное время, равное 29,81 мс. Эта особенность связана с работой Hibernate, поскольку он анализирует, а затем оптимизирует поступающий в него запрос на добавление [25]. После чего запоминает проделанное действие, чтобы дальнейшее, идентичное действие, происходило быстрее. Проанализировав полученные промежуточные результаты, можно рассчитать среднее время добавление практики для выбранного студента. Сложив полученное время экспериментов и поделив на их количество, получим среднее затраченное время, равное 9,903 мс.

Данный результат является хорошим, поскольку затраченное время настолько мало, что пользователь не получает дискомфорта от ожидания добавления. Это связано с оптимизацией sql-запроса, правильной связи сущностей в базе данных, хорошему соединению интернета.

Проведем следующий эксперимент, который должен выявить, является ли эффективным второй метод наполнения базы данных практик. Ниже, в таблице 3, представлены промежуточные результаты добавления записей.

Добавленные данные, метод измерения скорости используются из первого эксперимента.

Таблица 3 – Результаты замеров 2 метода добавления данных

| | 1 практика | 10 практик | 100 практик | 1000 практик |
|----|------------|------------|-------------|--------------|
| 1 | 32,07 мс | 84,77 мс | 330,28 мс | 2957,82 мс |
| 2 | 12,44 мс | 66,97 мс | 286,51 мс | 2860,74 мс |
| 3 | 10,06 мс | 63,96 мс | 278,84 мс | 2858,45 мс |
| 4 | 10,93 мс | 59,57 мс | 264,66 мс | 2855,902 мс |
| 5 | 9,48 мс | 58,03 мс | 281,3 мс | 2922 мс |
| 6 | 8,72 мс | 55,62 мс | 284,202 мс | 3022,22 мс |
| 7 | 9,01 мс | 58,76 мс | 275 мс | 2759,79 мс |
| 8 | 9,24 мс | 56,66 мс | 279,01 мс | 2783,46 мс |
| 9 | 9,18 мс | 54,69 мс | 285,343 мс | 2778,83 мс |
| 10 | 10,45 мс | 59,77 мс | 280,66 мс | 2802,88 мс |

Полученные данные были исследованы, и получены следующие выводы: при увеличении числа практик, добавляемых одновременно, сильно возрастает потраченное время. Проведенный эксперимент нам дает явно понять, что при увеличении количества добавляемых данных, увеличивается и время, затраченное на их наполнение в базу данных практик. Плюсом такого метода добавления практик, является возможность создавать записи для всей группы студентов непосредственно, не делая лишних движений. Такой метод позволяет преподавателю удостовериться что вся группа получит практику, которая ничем не будет отличаться от своих сокурсников [24].

Последний эксперимент будет проводиться на схожих с предыдущими замерами условиях. А именно: время, затраченное на добавление данных, будет замеряться с помощью функции `System.nanoTime()`, промежуточные результаты измерений будут представлены в идентичной таблице. Вводимые данные будут слегка отличаться и имеют вид, показанный на рисунке 20.

| name | vid | opisanie | start | finish | ocenka | sdano | familia | imya | otchestvo | gruppa |
|--------|----------|------------|------------|------------|--------|-------|-----------|-------|------------|--------|
| Крикет | Учебная | aaaaaaaa | 2023-11-12 | 2023-11-12 | 0 | 0 | Ильин | Керам | Витальевич | 888888 |
| Крикет | Ознаком | бббббббб | 2023-11-12 | 2023-11-13 | 1 | 0 | Сердюкова | Нина | Федоровна | 888888 |
| Крикет | Летняя | вввввввв | 2023-11-12 | 2023-11-14 | 2 | 1 | Сердюкова | Нина | Федоровна | 888888 |
| Крикет | Производ | гггггггггг | 2023-11-12 | 2023-11-15 | 3 | 1 | Сердюкова | Нина | Федоровна | 888888 |
| Крикет | Преддипл | дддддддд | 2023-11-12 | 2023-11-16 | 5 | 1 | Ильин | Керам | Витальевич | 888888 |
| Крикет | Учебная | aaaaaaaa | 2023-11-12 | 2023-11-12 | 0 | 0 | Ильин | Керам | Витальевич | 888888 |
| Крикет | Ознаком | бббббббб | 2023-11-12 | 2023-11-13 | 1 | 0 | Сердюкова | Нина | Федоровна | 888888 |
| Крикет | Летняя | вввввввв | 2023-11-12 | 2023-11-14 | 2 | 1 | Сердюкова | Нина | Федоровна | 888888 |
| Крикет | Производ | гггггггггг | 2023-11-12 | 2023-11-15 | 3 | 1 | Сердюкова | Нина | Федоровна | 888888 |
| Крикет | Преддипл | дддддддд | 2023-11-12 | 2023-11-16 | 5 | 1 | Ильин | Керам | Витальевич | 888888 |

Рисунок 20 – Импорт 10 записей

Полученные системой данные, были обработаны, отсортированы и добавлены в нашу базу данных. Они имеют следующий вид, представленный на рисунке 21.

| id | date_of_created | finish | name | oce | opisanie | sd | start | vid | stuc | tea |
|-------|----------------------------|------------|--------|-----|------------------|----|------------|------------------|------|-----|
| 11928 | 2023-06-11 15:11:44.113504 | 2023-11-12 | Крикет | 0.0 | aaaaaaaaaaaaaaaa | 0 | 2023-11-12 | Учебная | 1 | 1 |
| 11929 | 2023-06-11 15:11:44.116511 | 2023-11-13 | Крикет | 1.0 | бббббббббббббб | 0 | 2023-11-12 | Ознакомительная | 2 | 1 |
| 11930 | 2023-06-11 15:11:44.119506 | 2023-11-14 | Крикет | 2.0 | вввввввввввв | 1 | 2023-11-12 | Летняя | 2 | 1 |
| 11931 | 2023-06-11 15:11:44.122505 | 2023-11-15 | Крикет | 3.0 | гггггггггггггггг | 1 | 2023-11-12 | Производственная | 2 | 1 |
| 11932 | 2023-06-11 15:11:44.125497 | 2023-11-16 | Крикет | 5.0 | дддддддддддддд | 1 | 2023-11-12 | Преддипломная | 1 | 1 |
| 11933 | 2023-06-11 15:11:44.128734 | 2023-11-12 | Крикет | 0.0 | aaaaaaaaaaaaaaaa | 0 | 2023-11-12 | Учебная | 1 | 1 |
| 11934 | 2023-06-11 15:11:44.130906 | 2023-11-13 | Крикет | 1.0 | бббббббббббббб | 0 | 2023-11-12 | Ознакомительная | 2 | 1 |
| 11935 | 2023-06-11 15:11:44.138490 | 2023-11-14 | Крикет | 2.0 | вввввввввввв | 1 | 2023-11-12 | Летняя | 2 | 1 |
| 11936 | 2023-06-11 15:11:44.141483 | 2023-11-15 | Крикет | 3.0 | гггггггггггггггг | 1 | 2023-11-12 | Производственная | 2 | 1 |
| 11937 | 2023-06-11 15:11:44.144475 | 2023-11-16 | Крикет | 5.0 | дддддддддддддд | 1 | 2023-11-12 | Преддипломная | 1 | 1 |

Рисунок 21 – Добавленные 10 записей

Таблица 4 – Результаты замеров 3 метода добавления данных

| | 1 практика | 10 практик | 100 практик | 1000 практик |
|----|------------|------------|-------------|--------------|
| 1 | 43,7 мс | 81 мс | 357 мс | 3286,18 мс |
| 2 | 32 мс | 84 мс | 331,36 мс | 3196,93 мс |
| 3 | 42,38 мс | 69 мс | 350,24 мс | 3367,01 мс |
| 4 | 32,46 мс | 77,81 мс | 336,03 мс | 3221,66 мс |
| 5 | 31,79 мс | 66,02 мс | 329,31 мс | 3185,73 мс |
| 6 | 43,14 мс | 73,05 мс | 305,19 мс | 3161,88 мс |
| 7 | 30,09 мс | 82,93 мс | 301,58 мс | 3199,08 мс |
| 8 | 31,31 мс | 73,49 мс | 307,16 мс | 3147,56 мс |
| 9 | 42,99 мс | 71 мс | 307,04 мс | 3197,71 мс |
| 10 | 30,65 мс | 72,53 мс | 322,69 мс | 3169,14 мс |

Проведем анализ полученных данных и дадим полноценный ответ на вопрос, удовлетворяет ли полученный результат заявленным требованиям. Поскольку сам импорт данных состоит из нескольких частей и включает в себя прием файла, проверку на соответствие формату, формату данных, чтение данных и их обработку, запись в базу данных, то ожидается что этот метод будет неэффективен. Однако, согласно полученным результатам, можно сказать, что пользователь системы будет полностью удовлетворён скоростью работы системы, поскольку на запись 1000 строк ушло около 3200 мс. Такой результат является приемлемым и позволяет добавлять в систему наполнения базы данных практик большое количество информации одновременно, при этом не тратя большого числа времени.

Для того чтобы с уверенностью заявить об эффективности системы, нужно дополнить тестирование и сравнить средние результаты работы методов импорта данных и добавления для группы. Такое сравнение представлено в таблице 5. В ней были рассчитаны средние значения времени. Промежуточные данные были взяты из экспериментов, проведенных выше. Визуальное сравнение в виде диаграммы представлено на рисунке 22.

Таблица 5 – Средние значения времени 2 и 3 метода добавления данных

| | Добавление практики для группы студентов | Добавление практики путем импорта из файла |
|--------------|--|--|
| 1 практика | 12,158 мс | 36,132 мс |
| 10 практик | 61,88 мс | 75,083 мс |
| 100 практик | 284,505 мс | 324,76 мс |
| 1000 практик | 2760,209 мс | 3210,288 мс |

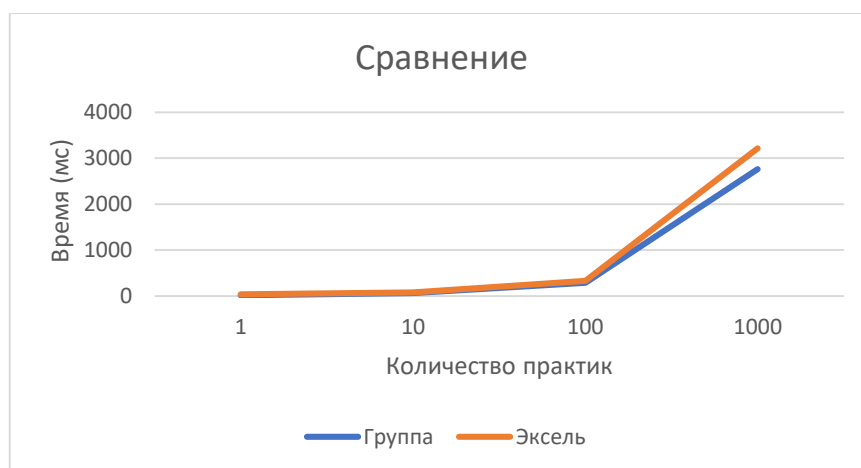


Рисунок 22 – Диаграмма сравнения

Разберем сравнение двух методов. Были подсчитаны средние значения времени, затраченного системой на внесение данных о практиках в БД. Согласно приведенным выше данным, чем больше введено информации, тем медленнее работает система. Так, например, при добавлении одной практики 2 методом, затрачено 12,158 мс, а при добавлении 10 практик тем же методом, затрачено 61,88 мс. Посчитав ускорение между этими промежутками, можно сказать, что оно равно 5, при увеличении количества строк с 1 до 10. Это является отличным результатом.

Также на представленном выше графике было проведено сравнение затраченного времени. Это связано с тем, что при таком добавлении не происходит полноценное, построчное считывание данных, которое происходит при импорте данных из сторонних источников.

Разобрав результаты проделанного тестирования, можно с уверенностью сказать, что все варианты добавления практик в систему наполнения базы данных практик соответствуют заявленным требованиям. Это соответствие выражается в комфорте использования этих методов, ведь пользователю необходимо быстро вносить информацию о практиках. Такое разнообразие способов положительно влияет на человека, использующего программу, и помогает ему эффективно добавлять практики в нашу систему.

Выводы по третьему разделу

В этом разделе были проделаны следующие шаги: разработан backend, разработаны элементы frontend, проведена сборка системы в Maven, проект протестирован и представлены результаты. Разработка backend происходила с использованием целого стека технологий и их взаимного действия. Именно благодаря им и навыкам программирования был написан основной функционал программы. Frontend-часть была разработана таким образом, чтобы соответствовать главным требованиям пользователя: быть понятной и удобной.

По итогу, практическим путем, было доказано, что проект является удобным для использования пользователем, не требующим специфичных навыков программирования и использования системы. Были проведены исследования и представлены результаты проделанных расчетов о времени, затраченном на добавления данных о практиках в разработанную систему. Также, с уверенностью можно сказать, что реализованная система безопасности полностью оправдывает ожидания и соответствует заявленным требованиям.

Заключение

Данная бакалаврская работа посвящена системе наполнения базы данных практик, заказчиком которой является отдел разработки информационных систем Тольяттинского государственного университета. В ходе выполнения работы были поставлен ряд задач на исследование.

Была рассмотрена используемая в ТГУ система наполнения базы данных практик, проанализированы ее преимущества и недостатки. Выделен основной функционал. В результате чего было принято решение о создании собственной системы наполнения базы данных практик.

Также были описаны методы добавления практик. После чего было принято решение использовать все выделенные методы в совокупности.

Были изучены предъявляемые функциональные требования, в результате чего было установлено что разрабатываемый продукт должен соответствовать всем современным нормам и стандартам качества, иметь ряд преимуществ над существующим аналогом.

Также была разработана система наполнения базы данных практик, основанная на предъявляемых функциональных требованиях. В ходе разработки были учтены интересы бизнес-процессов ТГУ.

Было проведено тестирование, которое затронуло не только backend-часть, но и часть frontend. В результате такого ручного тестирования, были выявлены недочеты и ошибки. В дальнейшем они все были устранены.

Задачи, определенные в ходе исследования, были выполнены в полном объеме. А именно: проанализирована используемая в ТГУ система наполнения, выявлены недостатки и проанализированы, описаны варианты их решения, определены главные требования к разрабатываемому программному продукту, определены методики и подходы, необходимые для решения выявленных недостатков, проанализирован рынок инструментов разработки программного обеспечения, выбраны необходимые инструменты разработки ПО, среда разработки, СУБД и т.д., разработана система

наполнения базы данных практик, отвечающая заявленным требованиям, разработана система тестирования проекта, проведена сборка и тестирование проекта, оценена эффективность созданной системы.

Практические испытания подтвердили удобство использования разработанного проекта для пользователей без специальных навыков программирования. Исследования показали, что время, затрачиваемое на добавление данных о практиках в новую систему, значительно сократилось. Кроме того, реализованная система безопасности полностью соответствует заявленным требованиям и обеспечивает надежную защиту данных, разнообразие методов ввода информации о практиках обеспечивает удобство и скорость работы пользователей. Благодаря этому, они могут легко и эффективно добавлять данные в систему, выбирая наиболее подходящий для себя способ.

Список используемой литературы и используемых источников

1. Алешин, Л.И. Информационные технологии: учебное пособие / Л.И. Алешин. – Москва : Маркет ДС, 2011. – 384 с.
2. Алиев, В.С. Информационные технологии и системы финансового менеджмента: учебное пособие / В.С. оглы Алиев. – Москва: Форум, ИНФРА-М, 2011. - 320 с.
3. Андерсен, В. Базы данных Microsoft Access. Проблемы и решения: Прост, пособ. / Пер. с англ. – Москва: Издательство ЭКОМ, 2001. – 384 с.
4. Архангельский, А.Я. Delphi 2006. Язык Delphi, классы, функции Win32 и NET: справочное пособие / А.Я. Архангельский. – Москва : Бином–Пресс, 2011. – 661 с.
5. Васильев, А. VBA в Office 2000: учебный курс/А. Васильев, А. Андреев. – Санкт–Петербург : 2001. – 432 с.
6. Ветитнев, А.М. Информационные технологии в социальнокультурном сервисе и туризме. Оргтехника: учебное пособие / А.М. Ветитнев. – Москва : Форум, 2010. – 400 с.
7. Вдовин, В.М. Информационные технологии в налогообложении: практикум / В.М. Вдовин, Л.Е. Суркова. – Москва: Дашков и К, 2012. – 248 с.
8. Вдовин, В.М. Информационные технологии в финансово-банковской сфере: Практикум / В.М. Вдовин. – Москва: Дашков и К, 2012. – 248 с.
9. Виллариал, Б. Программирование Access 2002 в примерах: Пер. с англ. – Москва: КУДИЦ – ОБРАЗ, 2003. – 496 с,
10. Гаврилов, М.В. Информатика и информационные технологии: учебник для бакалавров / М.В. Гаврилов, В.А. Климов; Рецензент Л.В. Кальянов, Н.М. Рыскин. – Москва: Юрайт, 2013. – 378 с.
11. Гарнаев, А.Ю. Самоучитель VBA /А.Ю. Гарнаев. – Санкт–Петербург: БХВ – Петербург, 2001. – 512 с.

12. Гвоздева, В.А. Информатика, автоматизированные информационные технологии и системы: учебник / В.А. Гвоздева. – Москва: ИД ФОРУМ, НИЦ ИНФРА–М, 2013. – 544 с.
13. Голицына, О.Л. Информационные технологии: учебник / О.Л. Голицына, Н.В. Максимов, Т.Л. Партыка, И.И. Попов. – Москва: Форум, ИНФРА–М, 2013. – 608 с.
14. Голубенко, Н.Б. Информационные технологии в библиотечном деле / Н.Б. Голубенко. – Ростов-на-Дону: Феникс, 2012. – 282 с.
15. Горев, А. Эффективная работа с СУБД / А. Горев, Р. Ахаян, С. Макашарипов. – Санкт–Петербург: Питер, 1997. – 704 с.
16. Гофман, В.Э. Работа с базами данных в Delphi / В.Э. Гофман, А.Д. Хомоненко. – Санкт–Петербург: БХВ–Петербург, 2006. – 656 с.
17. Грэм, Малкольм. Программирование для Microsoft SQL SERVER 2000 с использованием XML. Пер. с англ. М. Гаврилов, Л.П. Информационные технологии в коммерции: учебное пособие / Л.П. Гаврилов. – Москва: НИЦ ИНФРА-М, 2013. – 238 с.
18. Диго, С.М. Проектирование и использование баз данных / С.М. Диго. – Москва: Финансы и статистика, 1995. – 487 с.
19. Жданова, М.М. Вопросы формирования профессионально важных качеств инженера Вестник Таджикского технического университета 2011. Т. 4. № – 4. С. 122–124.
20. Желонкин, А. Основы программирования в интегрированной среде DELPHI: – Санкт–Петербург, Бином. Лаборатория знаний, 2004 г. – 240 с.
21. Apache FreeMarker, 2023 [Электронный ресурс]: офиц. сайт. URL: https://freemarker.apache.org/docs/ref_directive_if.html/, (дата обращения: 28.05.2023).
22. FreeMarker Common Operations, 2021 [Электронный ресурс]: информ. сайт. URL: <https://www.baeldung.com/freemarker-operations/>, (дата обращения: 28.05.2023).

23. Hibernate. Everything data, 2023 [Электронный ресурс]: офиц. сайт URL: <https://hibernate.org/>, (дата обращения: 16.05.2023).

24. IntelliJ IDEA, 2023 [Электронный ресурс]: офиц. сайт. URL: <https://www.jetbrains.com/idea/> (дата обращения: 26.05.2023).

25. Java Technology, 2023 [Электронный ресурс]: офиц. сайт. URL: https://www.java.com/download/faq/index_general.html/ (дата обращения: 25.05.2023).