МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Satbayev University

Институт информационных и информационных технологий

Кафедра кибербезопасность, обработка и хранение информации

Медеубеков Абылкаир Даниярович

Разработка Веб-приложения «Библиотека»

ДИПЛОМНАЯ РАБОТА

Специальность 5В070300 – Информационные системы

Алматы, 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН

Satbayev University

Институт кибернетики и информационных технологий

Кафедра кибербезопасность, обработка и хранение информации

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой КБОиХИ
канд. техн. наук, доцент
_____ Н. А. Сейлова
«____» _____20__г.

ДИПЛОМНАЯ РАБОТА

На тему: Разработка WEB-приложения «Библиотека»

Специальность 5В070300 – Информационные системы

Выполнил**:** Медеубеков. А. Д.

Научный руководитель

Магистр технических наук,тьютор
_____ Бауыржан М.Б.

«____» _____ 20__г.

Алматы, 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Satbayev University

Институт кибернетики и информационных технологий

Кафедра кибербезопасность, обработка и хранение информации

5В070300 – Информационные системы

УТВЕРЖДАЮ
Заведующий кафедрой КБОиХИ
канд. техн. наук, доцент
_____ Н. А. Сейлова
«____» _____20__г.

ЗАДАНИЕ
на выполнение дипломной работы

Обучающемуся: Медеубеков Абылкаир Даниярович
Тема: Разработка WEB-приложения «Библиотека»
Утверждена *приказом Ректора Университета № 762-п от «27» января 2020г.*
Срок сдачи законченной работы *«27» мая 2020г.*
Исходные данные к дипломному проекту: результаты преддипломной практики, результат обзора современного состояния по данной теме, сбор теоретического материала.
Краткое содержание дипломной работы:
а) Анализ существующих библиотек в сети интернет;
б) Программное и информационное обеспечение системы;
в) Разработка и тестирование;

Рекомендуемая основная литература: *из 11 наименований*

ГРАФИК

подготовки дипломной работы (проекта)

Наименование разделов, перечень разрабатываемых вопросов	**Сроки представления научному руководителю**	**Примечание**
Анализ существующих библиотек в сети интернет	27.02.2020г.	
Программное и информационное обеспечение системы	30.03.2020г.	
Разработка и тестирование	15.05.2020г.	

Подписи

консультантов и нормоконтролера на законченную дипломную работу (проект) с указанием относящихся к ним разделов работы (проекта)

Наименование разделов	**Консультанты, Ф.И.О. (уч.степень, звание)**	**Дата подписания**	**Подпись**
Разработка WEB-приложения «Библиотека»	Бауыржан М.Б., тьютор		
Нормоконтролер	Бауыржан М.Б., тьютор		
Программная часть	Кабдуллин М.А., ассистент		

Научный руководитель: _____ Бауыржан М.Б.

Задание принял к исполнению обучающийся _____ Медеубеков А.Д.

Дата "27" января 2020 г.

МИНИСТЕРСТВО НАУКИ И ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН
Университет САТБАЕВ

Отзыв научного руководителя

Дипломная работа

Медеубеков Абылкаир Даниярович

5В070300 – Информационные системы

Тема : Разработка WEB-приложения «Библиотека»

Дипломная работа содержит введение, заключение и список использованных литератур.

Данная дипломная работа связана с разработкой библиотеки электронных книг.

При разработке онлайн решения в учет шли уже существующие решения а так же были выведены их недостатки и положительные качества сервисов по обеспечению юным дарованиям необходимых знаний в исследуемой ими области. Любая библиотека является информационной системой, так как имеет доступ к базе данных. Разработанная информационная система отвечает потребностям рядового пользователя, может регистрировать новых пользователей, осуществлять поиск по заданному жанру и отображать соответствующие файлы. Разработанное веб-приложение использует следующие инструменты: Spring MVC(Model-View-Controller), Pg Admin4, Bootstrap.

Медеубеков Абылкаир в процессе выполнения дипломной работы показал умение сочетать теоретические знания и их практическое применение, зарекомендовал себя как самостоятельный студент.

Работа полностью отвечает требованиям, предьявленным к дипломным работам специальности “Информационные системы”, к защите допускается.

Научный руководитель:
Магистр технических наук,тьютор
_____М. Б. Бауыржан
«__» _____ 2020 г

Протокол анализа Отчета подобия Научным руководителем

Заявляю, что я ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

Автор: Медеубеков А.Д.

Название: Разработка Веб-приложения «Библиотека

Координатор:Мадина Бауыржан

Коэффициент подобия 1:6,3

Коэффициент подобия 2:2,4

Замена букв:2

Интервалы:0

Микропробелы:0

Белые знаки: 0

После анализа Отчета подобия констатирую следующее:

- ☐ обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, признаю работу самостоятельной и допускаю ее к защите;
- ☐ обнаруженные в работе заимствования не обладают признаками плагиата, но их чрезмерное количество вызывает сомнения в отношении ценности работы по существу и отсутствием самостоятельности ее автора. В связи с чем, работа должна быть вновь отредактирована с целью ограничения заимствований;
- ☐ обнаруженные в работе заимствования являются недобросовестными и обладают признаками плагиата, или в ней содержатся преднамеренные искажения текста, указывающие на попытки сокрытия недобросовестных заимствований. В связи с чем, не допускаю работу к защите.

Обоснование:

……

…… ……

Дата *Подпись Научного руководителя*

Протокол анализа Отчета подобия
заведующего кафедрой / начальника структурного подразделения

Заведующий кафедрой / начальник структурного подразделения заявляет, что ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

Автор: Медеубеков А.Д.

Название: Разработка Веб-приложения «Библиотека

Координатор:Мадина Бауыржан

Коэффициент подобия 1:6,3

Коэффициент подобия 2:2,4

Замена букв:2

Интервалы:0

Микропробелы:0

Белые знаки: 0

После анализа Отчета подобия констатирую следующее:

- ☐ обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, признаю работу самостоятельной и допускаю ее к защите;
- ☐ обнаруженные в работе заимствования не обладают признаками плагиата, но их чрезмерное количество вызывает сомнения в отношении ценности работы по существу и отсутствием самостоятельности ее автора. В связи с чем, работа должна быть вновь отредактирована с целью ограничения заимствований;
- ☐ обнаруженные в работе заимствования являются недобросовестными и обладают признаками плагиата, или в ней содержатся преднамеренные искажения текста, указывающие на попытки сокрытия недобросовестных заимствований. В связи с чем, не допускаю работу к защите.

Обоснование:

.....
.....
.....
.....
.....

....

Дата *Подпись заведующего кафедрой / начальника структурного подразделения*

Окончательное решение в отношении допуска к защите, включая обоснование:

.....
.....
.....
.....
.....

....

Дата *Подпись заведующего кафедрой /*

начальника структурного подразделения

StrikePlagiarism.com

ДАТА ОТЧЕТА: 2020-05-25 10:28:02

НАЗВАНИЕ:
Разработка Веб-приложения «Библиотека

АВТОР:
Медеубеков А.Д.

НАУЧНЫЙ РУКОВОДИТЕЛЬ:
Мадина Бауыржан

Показать детали:

Уровень заимствований

Обратите внимание!Высокие значения коэффициентов не означают плагиат. Отчет должен быть проанализирован экспертом.

6,30%	2,37%	0,83%
КП1	КП2	КЦ
25 Длина фразы для коэффициента подобия 2	**6131** Количество слов	**46751** Количество символов

Предупреждение и сигналы тревоги

В этом разделе вы найдете информацию, касающуюся манипуляций в тексте, с целью изменить результаты проверки. Для того, кто оценивает работу на бумажном носителе или в электронном формате, манипуляции могут быть невидимы (может быть также целенаправленное вписывание ошибок). Следует оценить, являются ли изменения преднамеренными или нет.

Замена букв	2	показать в тексте
Интервалы	0	показать в тексте
Микропробелы	0	показать в тексте
Белые знаки	0	показать в тексте

Заимствования по списку источников

Просмотрите список и проанализируйте, в особенности, те фрагменты, которые превышают КП №2 (выделенные жирным шрифтом). Используйте ссылку «Обозначить фрагмент» и посмотрите, являются ли выделенные фрагменты повторяющимися короткими фразами, разбросанными в документе (совпадающие сходства), многочисленными короткими фразами расположенные рядом друг с другом (парафразирование) или обширными фрагментами без указания источника ("криптоцитаты").

\- 10 самых длинных фраз (4,27 %)

Десять самых длинных фрагментов найденых во всех доступных ресурсах.

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ И АДРЕС ИСТОЧНИКА URL (НАЗВАНИЕ БАЗЫ)	АВТОР	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ		УДАЛИТЬ ВСЕ ОБОЗНАЧЕНИЯ
1	https://aws.amazon.com/ru/lambda/serverless-architectures-learn-more/		50	0,82 %	показать в тексте
2	https://aws.amazon.com/ru/lambda/serverless-architectures-learn-more/		35	0,57 %	показать в тексте
3	https://fb.ru/article/368693/potoki-java-sozdanie-i-zavershenie		34	0,55 %	показать в тексте
4	https://fb.ru/article/368693/potoki-java-sozdanie-i-zavershenie		26	0,42 %	показать в тексте
5	https://fb.ru/article/368693/potoki-java-sozdanie-i-zavershenie		24	0,39 %	показать в тексте
6	https://fb.ru/article/368693/potoki-java-sozdanie-i-zavershenie		22	0,36 %	показать в тексте
7	https://fb.ru/article/368693/potoki-java-sozdanie-i-zavershenie		20	0,33 %	показать в тексте
8	https://fb.ru/article/368693/potoki-java-sozdanie-i-zavershenie		18	0,29 %	показать в тексте
9	https://bootstrap-4.ru/docs/3.4/		18	0,29 %	показать в тексте
10	https://fb.ru/article/368693/potoki-java-sozdanie-i-zavershenie		15	0,24 %	показать в тексте

\+ из базы данных RefBooks (0,00 %)

\+ из домашней базы данных (0,00 %)

\+ из программы обмена базами данных (0,00 %)

\- из интернета (6,30 %)

Все фрагменты найдены в глобальных интернет-ресурсах открытого доступа.

АҢДАТПА

Бұл дипломдық жұмыс электронды онлайн кітапхананың құрастыруымен байланысты.

Интернеттегі шешімді әзірлеу кезінде бұрыннан бар шешімдер, сондай-ақ олардың кемшіліктері мен жас таланттарға өздері оқыған салада қажетті білім беру қызметтерінің жағымды қасиеттері ескерілді. Кез-келген электронды кітапхана ақпараттық жүйе болып табылады, өйткені оған мәліметтер базасы қол жетімді. Әзірленген ақпараттық жүйе қарапайым пайдаланушының қажеттіліктерін қанағаттандырады, жаңа пайдаланушыларды тіркей алады, қажет жанрды ескерте отырып, сәйкес файлдарды көрсете алады. Әзірленген веб-бағдарлама келесі құралдарды қолданады: Spring MVC (Model-View-Controller), PgAdmin4, Bootstrap.

АННОТАЦИЯ

Данная дипломная работа связана с разработкой библиотеки электронных книг.

При разработке онлайн решения в учет шли уже существующие решения а так же были выведены их недостатки и положительные качества сервисов по обеспечению юным дарованиям необходимых знаний в исследуемой ими области. Любая библиотека является информационной системой, так как имеет доступ к базе данных. Разработанная информационная система отвечает потребностям рядового пользователя, может регистрировать новых пользователей, осуществлять поиск по заданному жанру и отображать соответствующие файлы. Разработанное веб-приложение использует следующие инструменты: Spring MVC(Model-View-Controller), PgAdmin4, Bootstrap.

THE SUMMARY

This thesis is related to the development of an online library.

In process of developing an online solution, already existing solutions were taken into account, as well as their shortcomings and positive qualities of services to provide young talents with the necessary knowledge in their field of studying. Any online library is taken as an information system, as it has access to the database. The developed information system meets the needs of the average user, can register new users, search for a given genre and display the corresponding files. The developed web application uses the following tools: Spring MVC (Model-View-Controller), PgAdmin4, Bootstrap.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Знания – бесконечно разширяемой поток информации оседающий в разуме живых созданий, позволяющий проанализировать ситуацию в определенной сфере, заранее определить последовательность действий других людей или просто воспользоваться в необходимый момент в рабочей среде. Каждый человек дорожит ими, так как, даже если у него отнять все, что ему дорого, с ним всегда останутся его знания.

В эпоху информационных технолигий, когда глобальные индустрии переходят к оцифрованный, компьютеризированной индустрии нам с каждым днем открываются всё новые и новые горизонты, позволяющие черпать знания с любой точки земного шара. Все стало возможным с появлением сервисов удовлетворяющих поставленную цель пользователя сети интернет.

Сам интернет это неиссекаемый источник данных и информации, на первый взгляд, хаотично разбросанных на просторах интернета. При первом погружении в эту пучину данных, не ясно, в каком порядке стоит вести поиски, для этого, нашими помощниками выступают поисковые сервисы, предстающий перед нами в виде поисковой системы, служит она в качестве навигатора в интернет пространстве, на сайте или на площадке по размещению сериалов. С помощью поисковых систем нам нет необходимости преодолевать огромные расстояния до читальни или магазинов с книжными полками, чтобы потом долго искать необходимое чтиво и лишь затем начать поглощать знания, так необходимые для нашего разума для его развития. Теперь же книги могут быть легко найдены и все это благодаря электронным библиотекам, хранящими в своей базе данных терабайты учебных пособий, романов, научных работ и всего того, что осталось нам от наших предков в разные периоды времени, в разных цивилизациях. Даже сейчас люди слажено работают над оцифровкой древних книг для их дальнейшего сохранения потомкам.

Все больше и больше информации, используемой нами в повседневной жизни, поступает в электронном формате, тем самым, смещая традиционные печатные издания, которые уже переводятся в цифровой формат. Эта форма представления информации позволяет организовать работу с информацией и доступ к ней на совершенно ином уровне, не мыслимые ранее. Печатные издания сохраняют свою эстетическую ценность и значимость, но они более не являются единственным ресурсом передачи опыта и знаний в рукописном формате, иными словами, источником информации, хоть это и делает нас более зависимыми от электронных устройств, являющимися частью интернета вещей, но оспаривать их пользу для современного человека неуместно. Теперь же, обеспечением населения земли услугами по доступу к электронным книгам может заняться даже школьник или студент высшего учебного заведения.

1 Обзор и анализ существующих библиотек

1.1 Достоинства и преимущества электронных библиотек

Основной целью публичных библиотек является предоставление ресурсов и услуг в различных средствах массовой информации для удовлетворения потребностей отдельных лиц и групп в образовании, информации и личном развитии, включая отдых и досуг.

Электронная библиотека – хранилище, коллекция упорядоченных по тематике, по автору и названию электронных документов. Каждая электронная библиотека может вести поиск а также снабжать средствами навигации по автору, названию и тематики искомый книги в данной библиотеке. На каждом сайте который предоставляет доступ к онлайн библиотеки и её ресурсам находится регистрационная страница для новых пользователей, которые готовы соверашать определенные действия на данной площадке. Присутствует доступ к бесплатным книгам известных классиков, как Федор Достоевский, Александр Дюма, Антон Чехов, Ульям Шекспир и других именитых авторов.

В настоящее время академические библиотеки переходят от традиционных печатных ресурсов и превращаются в электронные хранилища. Этими ресурсами являются тезисы и диссертации, а также другие инновационные академические библиотеки.

Преимущество цифрового формата над бумажным представляется в увеличенном количестве выбора. В наши дни цифровые библиотеки предоставляют доступ к множеству материалов с потенциально бесконечным количеством ресурсов и выборок под рукой, в то время как ограничением для традиционных библиотек является физическое пространство, в котором людям приходится терять время на поиски, а сами книги занимают много места. С помощью интернета и облачных хранилищ возможности библиотек расширяются и предоставляют более широкие возможности, например, они служат долгосрочным хранилищем важных данных, исследований, информации и результатов экспериментов. За многовековую историю человечества из-за погодных условий, химических реакций проходящих в пергаментах, многими цивилизациями безвозвратно были утеряны части накопленных предками знаний, которые могли служить во благо человечества. Но сегодня, благодаря цифровым хранилищам прошлый опыт не повторится, поскольку онлайн-копии исследований, научных работ и культурных произведений могут быть сохранены для следующих поколений, создавая виртуальное наследие информации.

Благодаря технологиям поисковых систем - например, ранжированию или автоматическому расширению терминов - даже начинающие пользователи могут начать использовать цифровые библиотеки, выполняя поиск самостоятельно. Мгновенный доступ к поучительному контенту осуществим по мере доступа в интернет, пока доступно подключение к сети, цифровые библиотеки доступны в любом месте и в любое время с помощью

технологического устройства, такого как ПК, планшет или смартфон. Студенты могут просматривать онлайн-книги, изображения, видео и другие образовательные материалы без необходимости ждать и отправляться в ближайшую физическую библиотеку. Они могут делать это в формальной обстановке, будь то в школе, или дома, получая мгновенный доступ к необходимой информации. Хрупкие фотографии или древние документы должны противостоять нескольким передачам от рук в руки и консультациям в научном и культурном обществе, Цифровое хранение книг, и аудио, решает проблему ухудшения качества, с риском быть поврежденными. Благодаря оцифровке материалов можно получить доступ к содержимому, столько раз, сколько нужно студенту, используя соответствующие форматы (mp3, jpeg, pdf и т. д.), Которые определенно намного безопаснее использовать и не волноваться о повреждениях.

1.2 Особенности и отличия электронных библиотек

Электронные библиотеки могут отличаться доступом к материалам, хранящимся на сервере определенной площадки. Некоторые площадки предоставляют возможность читать онлайн тысячи книг за ежемесячную подписку, другие могут продавать каждую книгу-бестселлер поштучно. Оба варианта имеют место быть, в зависимости от авторских прав и наличия перевода определенного экземпляра.

Некоторые онлайн библиотеки имеют топы продаж или просто популярные книги среди читателей их сервиса, что наталкивает на мысль о присутствии в них системы оценок и рецензий от пользователей, зарегистрированных на сайте, которые уже прочли книгу. Так же могут разниться жанры книг и их обилие определенных авторов. За немалое количество времени ощутимое число сервиса перекочевало в аудио формат, и этим же способом предоставлять как бесплатные, так и платные аудиофайлы. Аудио формат может показаться даже удобнее бумажного, но все идет в сравнении и предпочтениях каждого.

1.3 Представители электронных библиотек

Одни из самых популярных сервисов на просторах СНГ по предоставлению зарубежной, образовательной и бизнес литературы. На примере двух онлайн библиотек, можно сделать вывод что почти все библиотеки подобные им имеют за собой как коммерческое, так и образовательное направление. Под «подобными», подразумеваются сайты и площадки по продаже книг или же доступ к ресурсам по подписке. Стоит принять во внимание, что не все библиотеки просят совершить перевод

денежных средств, что бы предоставить пользователю полный доступ к файлам хранимых на сервере.

Сайт сервиса «MyBook» выполнен в очень удобном стиле и отвечает критериям дружественного интерфейса, интуитивно можно найти необходимую кнопку и раздел с жанрами книг. Данный сервис предоставляет доступ к 246000 книгам и 46000 аудиокнигам, по премиум подписке, которые позволяет читать онлайн весь каталог электронных файлов. При желании сервис стабильно функционирует на мобильных платформах IOS (версии свыше 10) и Android (версий выше 4.1) архитектурах. Для ознакомления с интересующими вас книгами на сайте присутствует пробный период и действует он в течении 14 дней, то есть, у вас ровно 2 недели для полного погружения. При необходимости можно загрузить интересующую нас книгу себе на устройство, и читать даже без доступа к интернету, сайт mybook.ru и мобильное приложение работают как одно целое, все библиотека данных обновляется ежеминутно, синхронизируя данные вашего аккаунта. Это позволяет читателю начать книгу на одном устройстве по пути на работу, а продолжить уже с планшета или же персонального компьютера, все заметки, выделенные цитаты и заметки так же переносятся на все перечисленные устройства, обеспечивая комфорт на любом подручном устройстве. Не стоит забывать, что мотивация при чтении позволяет нам впитать как можно больше информации. Сервис ведет статистику о вашем процессе чтения, ежемесячно отправляя на почту статистику процесса за весь месяц. Система также имеет подборки, подсказывая, что сейчас модно среди читателей, или малоизвестно, что почитать во время зимы, лета или во время карантина. Группа компаний «ЛитРес» являются владельцами сервиса «MyBook».

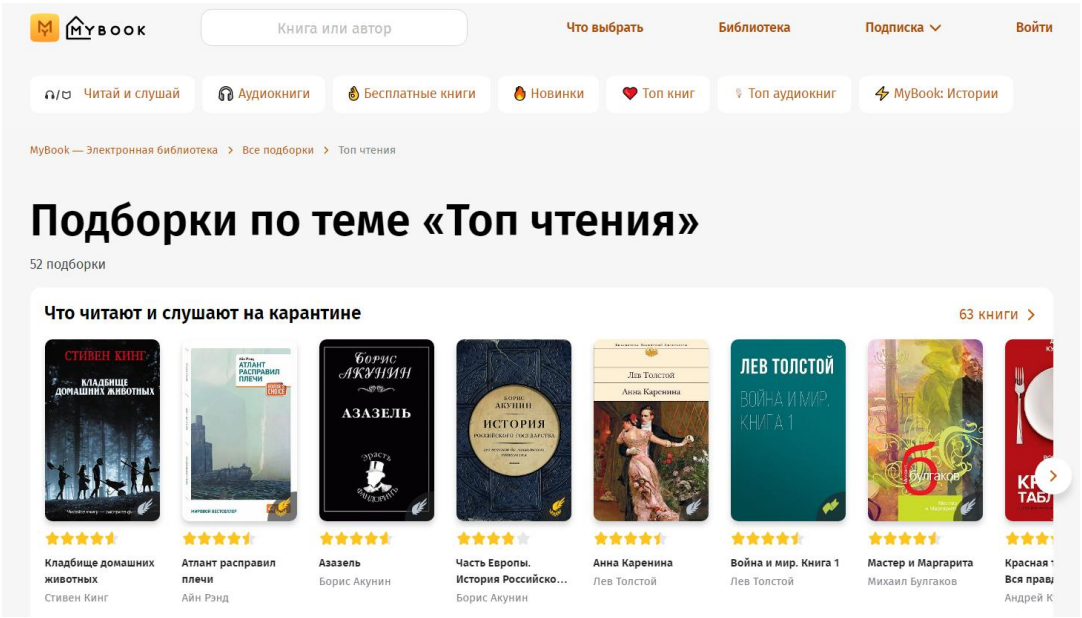

Рисунок 1.1 – Интерфейс сайта «MyBook»

Касательно сервиса «ЛитРес», Ассортимент данного сервиса насчитывает порядка 1 млн книг на русском и иностранных языках, 48 тысяч из которых доступны к бесплатному прочтению, на данной площадке часто проходят акции и скидки, позволяющие сэкономить на покупке книг. Купленную книгу можно скачать в форматах: FB2, EPUB, PD, MOBI, TXT и других. При покупке электронной книги она останется на вашем аккаунте. Ежемесячно каталог пополняется на 5 тысяч произведений. Положительной стороной является предоставление бесплатного фрагмента для ознакомления с книгой, который составляет четверть от самой книги. Основные преимущества не сильно разнятся от вышеупомянутого сервиса «MyBook».

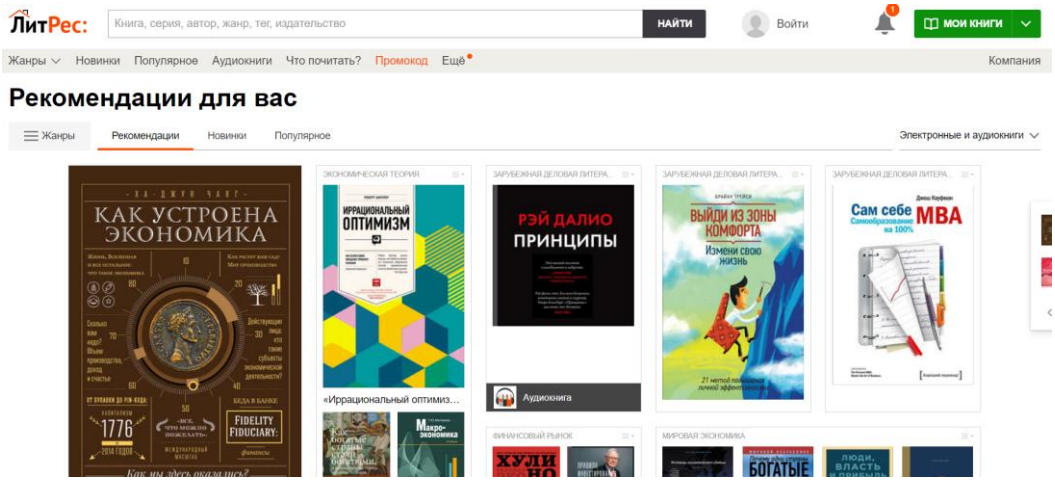

Рисунок 1.2 – Интерфейс сайта «ЛитРес»

1.4 Образовательные порталы с доступом к учебному материалу

В сети интернет можно найти не только платные книги с большой наценкой, но и бесплатные сайты, всячески пытающиеся подтолкнуть нас к саморазвитию. Ими могут послужить сайты с бесплатными книгами и образовательные порталы с учебниками по сферам: экономики, бизнеса, самодисциплины и математики, таких учебников много и человек легко найдет на таких сайтах что-нибудь полезное.

В распоряжении многих учебных заведений присутствует своя онлайн библиотека. Электронные библиотеки - лучший инструмент для предоставления онлайн-ресурсов для исследований, которые будут удобны для пользователей. Ретроспективный поиск прост и удобен, чем печатный ресурс. В таких библиотеках осуществим поиск по комбинации ключевых слов, тем самым электронные библиотеки облегчают своим читателям поиск решения. Так же, плюсом является предоставление для распечатки и сохранения загруженных элементов для будущих ссылок.

1.5 Постановка задачи

Основной задачей данной дипломной работы является разработка веб-приложения «Comelot», представляющее из себя систему с аутентификацией и авторизацией пользователей. В базе данных приложения должны находится данные о пользователях, и распространяемые ими файлы.

Постановка задачи:

- разработка базы данных пользователей;
- авторизация и аутентификация пользователей;
- разработка интерфейса приложения;
- загрузка файлов от имени пользователей;
- поиск файлов по метке.

2 Программное и информационное обеспечение системы

2.1 Инструменты разработки

В зависимости от выбранного язык программирования выбираются соответствующие инструменты, выбрав язык Java, можно использовать ряд инструментов облегчающих сборку проекта. Java – широко используемый язык программирования. Это самый популярный язык программирования используемый для написания приложений для устройств поддерживаемых операционной системой android. Java разработан, чтобы иметь схожий вид с языком программирования C++, но он проще в использовании, обеспечивает реализацию модели объектно-ориентированного программирования. Java используется для создания законченных приложений, которые могут работать на одном компьютере или распределяться между серверами и клиентами в сети. Трудно представить единственную причину того, почему язык программирования Java стал настолько повсеместным. Программы, созданные на этом языке программирования, предлагают переносимость в сети. Исходный код компилируется в то, что называют байт-кодом, который может выполняться в любом месте сети, на сервере или на клиенте с виртуальной машиной Java (JVM). Виртуальная машина интерпретирует байт-код в код, который будет работать на соответствующем компьютерном оборудовании. Именно виртуальная машина Java позволяет создавать кроссплатформенные решения.

Java является объектно-ориентированным. Объекты состояи из данных в виде полей или атрибутов и кода в виде процедур или методов. Объект может быть частью класса объектов, чтобы наследовать код, общий для класса. Метод - это возможности поведение объекта. Данный язык был построен в основном как объектно-ориентированный язык поскольку на дизайн Java оказал влияние тот же C ++. В нем также используется автоматический сборщик мусора для управления жизненными циклами объектов. Программист будет создавать объекты, но автоматический сборщик мусора восстановит память, как только объект больше не будет использоваться. Однако утечки памяти могут возникать, когда объект, который больше не используется, хранится в контейнере. Код написанный на языке Java надежнее чем в программах написанных на C++. Так как Java не содержат ссылок на данные, внешние по отношению к себе или другим известным объектам. Это гарантирует, что инструкция не может включать адрес данных, хранящихся в другом приложении или в самой операционной системе, что может привести к утечке данных или аварийному завершению. Еще одной особенностью можно выделить безопасность данных. В отличие от того же C ++, Java не использует указатели, которые могут быть незащищенными. Данные, преобразованные в байт-код с помощью этого языка программирования, также не читаются людьми. Кроме того, Java будет запускать программы внутри песочницы, чтобы предотвратить изменения из неизвестных источников. JVM выполняет ряд

проверок для каждого объекта для обеспечения целостности. Помимо того, что Java-апплет выполняется не на сервере, а на клиенте, он обладает и другими характеристиками, предназначенными для быстрой работы, что повышает гибкость разработки. Разработчики могут быстро освоить Java. С синтаксисом, подобным С ++, Java относительно легкий в изучении, особенно для тех, кто имеет опыт работы с С.

Основные платформы на которых разрабатываются приложения:

– Java SE - простые, автономные приложения, разработанные с использованием Java Standard Edition.

– Java EE - Java Enterprise Edition, предоставляет возможность создавать Java-программы, которые могут взаимодействовать с интернет-клиентами, включая веб-браузеры и веб-службы на основе REST и SOAP.

– Java ME - Java также предоставляет облегченную платформу для мобильной разработки, известную как Java Micro Edition.

Разработчикам легко писать программы, в которых используются популярные шаблоны проектирования и лучшие практики, используя различные компоненты, имеющиеся в Java EE. Например, такие инфраструктуры, как Struts и JavaServer Faces, используют сервлет Java для реализации шаблона проектирования фронт-контроллера для централизации запросов. Большая часть экосистемы Java - это большое разнообразие проектов с открытым исходным кодом и проектов сообщества, программных платформ и API. Среды Java EE также можно использовать в облаке. Разработчики могут создавать, развертывать, отлаживать и отслеживать Java-приложения, как пример, в Microsoft Azure на масштабируемом уровне. Java обычно используется в качестве языка программирования для приложений Android. Разработчики Android предпочитают Java из-за безопасности Java, объектно-ориентированных парадигм, регулярно обновляемых и поддерживаемых наборов функций, использования JVM и сред для сетей, ввода-вывода и потоков. Несмотря на то, что Java широко используется, она все еще подвергается серьезной критике. Синтаксис Java часто критикуют за то, что он слишком многословен. В ответ появилось несколько периферийных языков для решения этих проблем, включая Groovy. Благодаря тому, что Java ссылается на объекты внутри, сложные и параллельные операции на основе списка замедляют работу JVM. Язык Scala устраняет многие недостатки языка Java, которые ограничивают его возможности масштабирования.

В качестве инструментов разработки были выбраны несколько высокоэффективных инструментов такие как: Spring MVC, PgAdmin, Bootstrap.
В качестве паттерна проектирования был выбран Spring MVC. MVC расшифровывается как: Model, View, Controller. Данный тип архитектуры приложения состоит из трех блоков, где Модель (Model) отвечает за представление данных, относится к полному управлению данными и представляет собой Java-объекты. Вид (View) – внешнее представление, отвечает за отображение данных модели пользователя и реагирует на изменение моделей. Контроллер (Controller) – отвечает за получение данных,

обрабатывает запросы пользователя, создает соответствующую модель и передает её в вид.

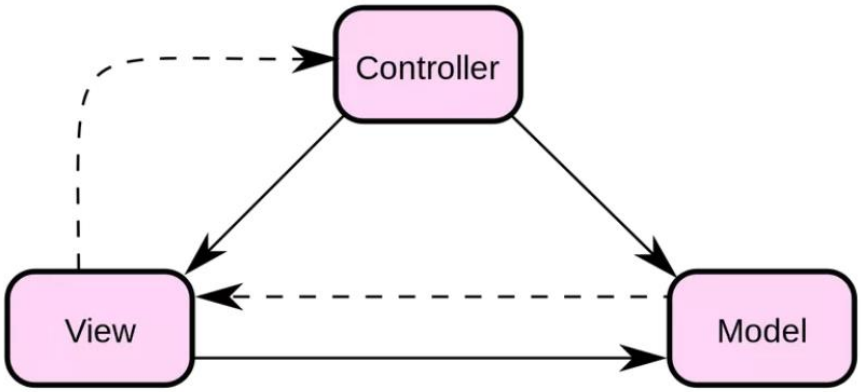

Рисунок 2.1 – Принцип работы MVC

Фреймворк Spring MVC обеспечивает архитектуру данного паттерна при помощи отдельно написанных компонентов, при этом обеспечивает свободную связь между ними, разделяя бизнес-логику, UI и логику ввода.

Вокруг DispatcherServlet строится все логика Spring MVC. DispatcherServle обрабатывает входящий запрос вида HTTP Request, и обращается к Handler Mapping, тем самым определяя необходимы контроллер и направляет его в нужный экземпляр. Принимающий контроллер на основании метода GET или POST, вызывает метод, который определяет данные модели, обрабатывает этот запрос в соответствии с настроенными интерфейсами, выдает HTTP Response, отправляя данные модели в вид, отображаемых в браузере.

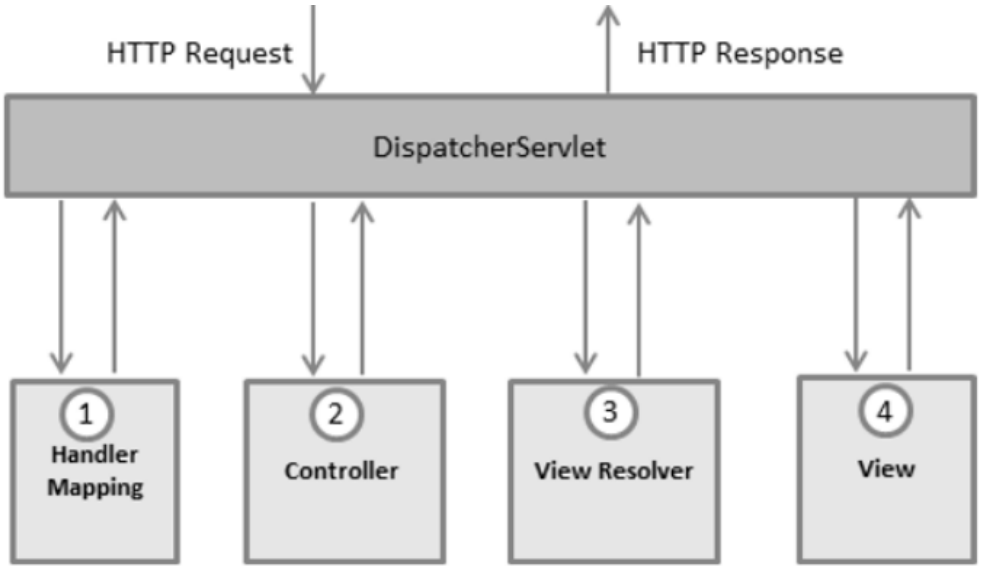

Рисунок 2.2 – Последовательность событий при запросе MVC

В качестве сборщика проектов был использован инструмент Maven, который используется для сборки проектов на языке Java. Maven был создан для облегчения команд для сборки разрабатываемого ПО, поскольку сложность сборки повышается, при обильном использовании сторонних библиотек и ресурсов. Особенность данного фреймворка – декларативное описание проекта, что позволяет настроить необходимые параметры проекта заранее, при желании их можно изменить.

База данных – это система, в которой мы можем хранить наши данные, есть возможность получить данные из него, управляйте этими же данными. Эти системы в основном обслуживаются серверами в базе данных. Каждое приложение имеет свою базу данных. PostgreSQL поддерживает много типов данных, таких как строковые, числовые, дата и время, такие как My SQL. Кроме того, он поддерживает типы данных для геометрических фигур, изображений, сетевых адресов, битовых строк, текстового поиска и записей JSON.

PostgreSQL – это программное обеспечение с открытым исходным кодом для реляционных баз данных, которое работает на платформе Linux и формирует с объектами в качестве реляционного компонента в системе управления базами данных. Он использует язык структурированных запросов для доступа к данным в таблицах базы данных, и поэтому он также называется Postgres. Некоторые из характерных особенностей этой базы данных: она очень надежна, процесс восстановления не требует усилий, а затраты на обслуживание очень малы. Это объектно-реляционная система управления базами данных, и она использует SQL (язык структурированных запросов) в качестве основного языка запросов.

Пользовательский интерфейс – это графическое оформление приложения, соединение между пользователем и компьютерной программой. Он состоит из кнопок, которые нажимают пользователи, текста, который они читают, изображений, ползунков, полей ввода текста и всех остальных элементов, с которыми взаимодействует пользователь. Интерфейс – это набор команд или меню, через которые пользователь общается с программой. Управляемый командами интерфейс - это интерфейс, в котором вы вводите команды. Управляемый меню интерфейс - это интерфейс, в котором вы выбираете команды из различных меню, отображаемых на экране. Пользовательский интерфейс является одной из наиболее важных частей любой программы, поскольку он определяет, насколько легко вы можете заставить программу делать то, что вы хотите. Мощная программа с плохо разработанным пользовательским интерфейсом не имеет большого значения. Графические пользовательские интерфейсы (GUI), использующие окна, значки и всплывающие меню, стали стандартными для персональных компьютеров.

Bootstrap - это самая популярная среда с открытым исходным кодом HTML, CSS и JavaScript (изначально созданная Twitter), которую вы можете использовать в качестве основы для создания веб-сайтов или веб-приложений.

Bootstrap делает разработку веб-интерфейса быстрее и проще. Это сделано для людей всех уровней квалификации, устройств всех форм и проектов всех размеров. Данный фреймворк поставляется с CSS, но в его исходном коде используются два самых популярных CSS-препроцессора, Less и Sass. Быстро начать работу с предварительно скомпилированным CSS или создать исходный код. Он легко и эффективно масштабирует ваши веб-сайты и приложения с помощью единой базы кода, от телефонов до планшетов и настольных компьютеров с медиа запросами CSS. С Bootstrap разработчик получает обширную и красивую документацию по общим элементам HTML, десяткам пользовательских компонентов HTML и CSS и удивительным плагинам jQuery.

2.2 Архитектура приложения

Архитектура приложений - это процесс определения структуры прикладных решений в соответствии с требованиями бизнеса или поставленной задачи. Он включает определение ландшафта приложения с целью оптимизации этого ландшафта в соответствии с идеальным планом. Архитектура приложений покрывает достаточно широкую область, которая начинается с идентификации того, какие прикладные системы нужны предприятию для выполнения бизнес-процессов, и включает такие аспекты, как проектирование, разработка и интеграция прикладных систем.

Для создания крупного проекта необходимы вычислительные ресурсы, хорошим выбором может служить бессерверная архитектура. Бессерверные архитектуры - это проекты приложений, которые включают сторонние сервисы «Backend as a Service» (BaaS) или включают в себя пользовательский код, запускаемый в управляемых контейнерах на платформе «Functions as a Service» (FaaS). Является частью модели облачных вычислений. Используя эти идеи и связанные с ними, такие как одностраничные приложения, такие архитектуры устраняют большую часть потребности в традиционном компоненте сервера, который всегда включен. Бессерверные архитектуры могут выиграть от значительного снижения эксплуатационных расходов, сложности и времени разработки, в связи с этим была выбрана данная архитектура, способ создания и запуска приложений и сервисов без необходимости управления инфраструктурой может быть использована в случае расширения и официального запуска приложения в сеть. Приложение по-прежнему работает на серверах, но управление этими серверами AWS полностью берет на себя. Это избавляет от необходимости заниматься выделением ресурсов, масштабированием и обслуживанием серверов для запуска приложений, баз данных и систем хранения данных. При использовании данной архитектуры разработчики могут сосредоточиться на основном продукте, не заботясь об управлении серверами или средами исполнения и об их обслуживании при работе как в облаке, так в локальной среде. Это позволяет разработчикам

экономить время и силы, которые можно потратить на разработку отличных продуктов с высокой надежностью и возможностью масштабирования.

Впервые такой сервис был использован для описания приложений, которые в значительной степени или полностью включают сторонние облачные приложения и службы для управления логикой и состоянием на стороне сервера. Как правило, это приложения «богатого клиента» - например, одностраничные веб-приложения или мобильные приложения, - которые используют обширную экосистему баз данных, доступных в облаке (например, Parse, Firebase), службы аутентификации и другие.

Безсерверность также может означать приложения, в которых логика на стороне сервера все еще пишется разработчиком приложения, но, в отличие от традиционных архитектур, она запускается в вычислительных контейнерах без сохранения состояния, которые запускаются событиями, эфемерны (могут длиться только один вызов) и полностью управляются третья сторона. Один из способов думать об этом - «Функции как услуга» или «FaaS». В настоящее время AWS Lambda является одной из самых популярных реализаций платформы Functions-as-a-Service, но существует множество других.

BaaS и FaaS связаны по своим эксплуатационным атрибутам (например, без управления ресурсами) и часто используются вместе. Все крупные поставщики облачных услуг имеют «портфели без серверов», которые включают в себя как продукты BaaS, так и продукты FaaS, например, на странице продуктов Amazon для серверов без сервера. База данных Google Firebase BaaS имеет явную поддержку FaaS через облачные функции Google для Firebase.

Существует аналогичная связь двух областей от небольших компаний. Auth0 начал с продукта BaaS, в котором реализовано множество аспектов управления пользователями, а затем создал сопутствующий сервис FaaS Webtask. Компания продвинула эту идею еще больше с Extend, которая позволяет другим компаниям SaaS и BaaS легко добавить возможность FaaS к существующим продуктам, чтобы они могли создать унифицированный продукт без сервера.

Традиционно архитектура будет выглядеть примерно так, как показано на рисунке ниже. Допустим, он реализован в Java или Javascript на стороне сервера с компонентом HTML + Javascript в качестве клиента.

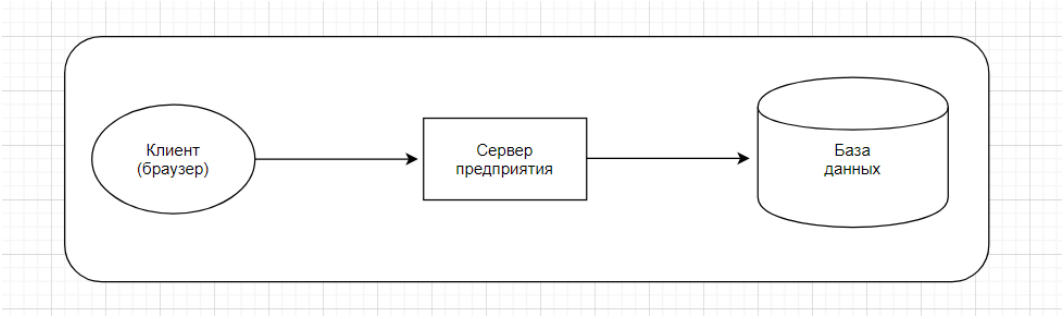

Рисунок 2.3 – Традиционная архитектура

С этой архитектурой клиент может использовать ресурсы не эффективно, так как большая часть логики в системе - аутентификация, навигация по страницам, поиск, транзакции - реализуется серверным приложением.

С архитектурой без сервера это выглядит иначе.

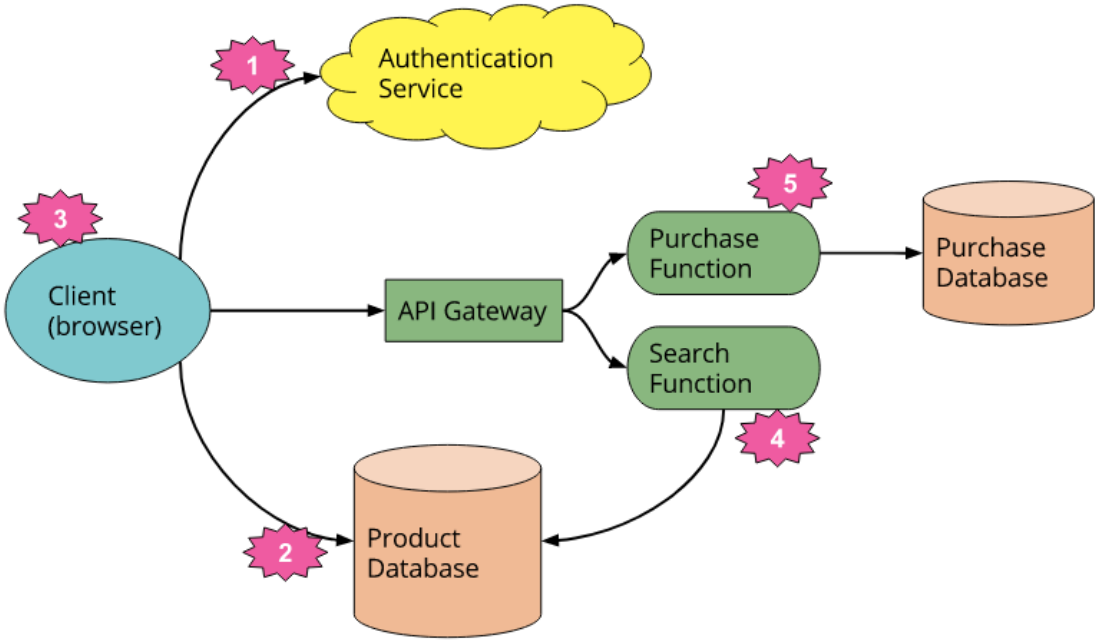

Рисунок 2.4 – Архитектура без сервера

Ранее уже упоминалась про FaaS, если подробнее, FaaS не требуют написания кода для конкретной платформы или библиотеки. Функции FaaS являются обычными приложениями, когда дело касается языка и среды. Например, функции AWS Lambda могут быть реализованы «первым классом» в Javascript, Python, Go, любом языке JVM или любом языке .NET. Однако ваша лямбда-функция также может выполнять другой процесс, связанный с ее артефактом развертывания, так что есть возможность использовать любой язык, который можно скомпилировать в процесс Unix. Однако функции FaaS имеют значительные архитектурные ограничения, особенно когда речь идет о состоянии и продолжительности выполнения. Развертывание сильно отличается от традиционных систем, поскольку отсутствуют серверные приложения для запуска самих себя. В среде FaaS загружают код для функции поставщику FaaS, и поставщик делает все остальное, необходимое для предоставления ресурсов, создания виртуальных машин, управления процессами и т.д.

Одним из аспектов бессерверной архитектуры, которое было показона ранее, является «шлюз API». Шлюз API - это HTTP-сервер, в котором маршруты и конечные точки определены в конфигурации, и каждый маршрут связан с ресурсом для обработки этого маршрута. В безсерверной архитектуре такие обработчики часто являются функциями FaaS. Когда шлюз API получает

запрос, он находит конфигурацию маршрутизации, соответствующую запросу, и, в случае маршрута, поддерживаемого FaaS, вызывает соответствующую функцию FaaS с представлением исходного запроса. Обычно шлюз API позволяет сопоставлять параметры запроса HTTP с более кратким вводом для функции FaaS или позволяет передавать весь запрос HTTP, как правило, в виде объекта JSON. Функция FaaS выполнит свою логику и вернет результат в шлюз API, который, в свою очередь, преобразует этот результат в ответ HTTP, который он передает обратно исходному вызывающему. У веб-сервисов Amazon есть собственный API-шлюз (слегка озадаченно называемый «API-шлюз»), и другие поставщики предлагают аналогичные возможности. Amazon API Gateway - это услуга BaaS. Сама по себе, поскольку она является внешней службой, которую настраивают разработчики, но не нуждаетесь в том, чтобы ее запускать или предоставлять самостоятельно. Помимо чисто маршрутизации запросов, шлюзы API также могут выполнять аутентификацию, проверку входных данных, сопоставление кода ответа и многое другое. Одним из вариантов использования API-шлюза с функциями FaaS является создание микросервисов на основе HTTP без использования сервера со всеми возможностями масштабирования, управления и другими преимуществами, которые получают функции FaaS.

Безсерверность – самое простое решение для аутсорсинга. Он позволяет платить кому-то за управление серверами, базами данных и даже за логику приложения, которой могли бы управлять владельцы самостоятельно. Поскольку используется предопределенный сервис, которым будут пользоваться многие другие люди, видим эффект экономии масштаба: владелец платит меньше за управляемую базу данных, потому что один поставщик использует тысячи очень похожих баз данных Сокращенные затраты представляются как совокупность двух аспектов. Во-первых, это рост стоимости инфраструктуры, который достигается исключительно за счет совместного использования инфраструктуры (например, аппаратного обеспечения, сетей) с другими людьми. Во-вторых, это увеличение затрат на рабочую силу: владелец сможете тратить меньше своего времени на аутсорсинговую систему без сервера, чем на эквивалент, разработанный и размещенный им же. Однако это преимущество не слишком отличается от того, что владелец получает от инфраструктуры как услуги (IaaS) или платформы как услуги (PaaS). Но можно расширить это преимущество двумя ключевыми способами, по одному для каждого из безсерверных BaaS и FaaS.

IaaS и PaaS основаны на предпосылке, что управление сервером и операционной системой может быть объединено. С другой стороны, безсерверный Backend as a service является результатом реализации всех компонентов приложения. Аутентификация является хорошим примером. Многие приложения кодируют свои собственные функции аутентификации, которые часто включают такие функции, как регистрация, вход в систему, управление паролями и интеграция с другими поставщиками аутентификации. В целом, эта логика очень похожа для большинства приложений, и такие

сервисы, как Auth0, были созданы, чтобы позволить нам интегрировать готовые функции аутентификации в приложение без необходимости разрабатывать его самостоятельно. В том же потоке находятся базы данных BaaS, такие как служба баз данных Firebase. Некоторые команды мобильных приложений считают целесообразным, чтобы клиент напрямую связывался с базой данных на стороне сервера. База данных BaaS устраняет большую часть накладных расходов на администрирование базы данных и, как правило, предоставляет механизмы для выполнения соответствующей авторизации для различных типов пользователей в шаблонах, ожидаемых от приложения без сервера.

2.3 Создание базы данных

Для нужд современных динамических веб-приложений доступ к базе данных обязателен. Имея базу данных для сайта, можно сделать ее более динамичной, содержательной и информативной. И когда дело доходит до высочайшего уровня безопасности и стабильности программного обеспечения, ответом является PostgreSQL. Сервер PostgreSQL - это решение корпоративного класса с открытым исходным кодом, позволяющее создавать огромные базы данных и предлагающее функциональные возможности и характеристики, доступные только на дорогих корпоративных серверах SQL (Oracle, MsSQL). Основанный на очень популярном программном обеспечении для баз данных Ingres, PostgreSQL является одной из самых безопасных и стабильных баз данных с открытым исходным кодом. Доказательством его стабильности и надежности является тот факт, что базы данных PostgreSQL в настоящее время используются многими крупными компаниями.

Чтобы иметь веб-сайт, который работает с информацией, наиболее распространенным подходом является сохранение этой информации в базе данных. Программные средства базы данных будут хранить эту информацию надежно и безопасно, готовые отображать ее по запросу. На примере обычного форума можно рассмотреть функции, все сообщения и вся информация для разных пользователей хранятся в соответствующих базах данных. Это также дает вам серьезное преимущество в бэкэнде веб-сайта - код, обеспечивающий его работу, может быть очень простым, всего за несколько запросов к базе данных, чтобы получить всю необходимую информацию. Как и следует из названия «программное обеспечение для баз данных», PostgreSQL предназначен для работы с базами данных и работы с ними. База данных состоит из таблиц, которые предназначены для хранения информации. Таблица в базе данных, как и графическая таблица, состоит из столбцов и строк, в которых столбцы обозначают тип хранимой в них информации, а строки содержат фактическую информацию.

Есть несколько способов создать базу данных в PostgreSQL. Один из самых простых - через командную строку PostgreSQL. Там вы просто должны набрать команду «CREATE DATABASE» и указать имя базы данных. Помните,

что вы должны находиться в режиме «SUPERUSER», чтобы создавать новые базы данных из командной строки. Создав базу данных, присутствует возможность перейти к ней и добавить в нее таблицы, в которых можно хранить необходимую информацию. Это также очень легко сделать через командную строку. Если разработчик предпочитает более графический интерфейс для создания базы данных, он должен проверить панель управления хостингом NTC. В нем можно использовать меню баз данных PostgreSQL, чтобы легко создать новую базу данных всего за несколько простых шагов. Все, что нужно сделать, это просто указать имя базы данных и пароль. Затем нажмите кнопку «Добавить базу данных PgSQL», чтобы создать новую базу данных.

После создания есть возможность легкого управлять базой данных с помощью программного обеспечения PgAdmin. Чтобы получить к нему доступ, все, что нужно сделать, это щелкнуть значок рядом с именем базы данных. Оказавшись внутри PgAdmin, можно легко управлять своей базой данных, создавать таблицы и заполнять их необходимой информацией. Нет необходимости ждать, пока база данных станет доступной - как только вы ее создадите, вы сможете начать управлять ею. Для удовлетворения потребностей своих клиентов NTC Hosting предлагает PostgreSQL со своими тарифными планами веб-хостинга Plus, Value и Exclusive, а также в качестве дополнительной услуги для остальных. Все веб-серверы специально оптимизированы для поддержки PostgreSQL, чтобы предоставить клиентам все инструменты, необходимые для поддержания и поддержания веб-сайта самого высокого качества.

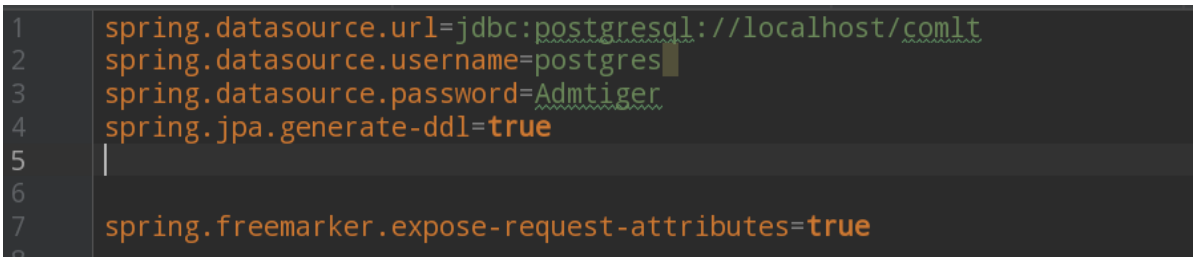

Рисунок 2.5 – Подключение базы данных в Spring

Структуру базы данных формирует JPA (Java Persistence API). Отображение объектов Java в таблицы базы данных и наоборот называется объектно-реляционным отображением (ORM). Java Persistence API является одним из возможных подходов к ORM. С помощью JPA разработчик может отображать, хранить, обновлять и извлекать данные из реляционных баз данных в объекты Java и наоборот. JPA можно использовать в приложениях Java-EE и Java-SE.

JPA является спецификацией и доступно несколько реализаций. Популярными реализациями являются Hibernate. Отображение между объектами Java и таблицами базы данных определяется с помощью метаданных

постоянства. Метаданные JPA обычно определяются с помощью аннотаций в классе Java.

```
<!--Database-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <scope>runtime</scope>
</dependency>
```

Рисунок 2.6 – Добавление поддержки Spring JPA в Maven

Класс, который должен быть сохранен в базе данных, должен быть аннотирован javax.persistence.Entity. Такой класс называется Entity. JPA использует таблицу базы данных для каждой сущности. Сохраненные экземпляры класса будут представлены одной строкой в таблице. Все классы сущностей должны определять первичный ключ, должны иметь не аргументный конструктор и не должны быть окончательными. Ключи могут быть одним полем или комбинацией полей. JPA позволяет автоматически генерировать первичный ключ в базе данных с помощью аннотации @GeneratedValue. JPA позволяет определять отношения между классами, например, можно определить, что класс является частью другого класса (содержания). Классы могут иметь отношения один к одному, один ко многим, многие к одному и многие ко многим с другими классами. Отношения могут быть двунаправленными или однонаправленными, например, в двунаправленном отношении оба класса хранят ссылку друг на друга, в то время как в однонаправленном случае только один класс имеет ссылку на другой класс.

```
1  package com.example.comlt.domain;
2  import javax.persistence.*;
3
4  @Entity
5  public class Message {
6      @Id
7      @GeneratedValue(strategy=GenerationType.AUTO)
8      private Integer id;
9
10     private String text;
11     private String tag;
12
13     @ManyToOne(fetch = FetchType.EAGER)
14     @JoinColumn(name = "user_id")
15     private User author;
```

Рисунок 2.7 – Использование JPA в приложении

Использование Spring значительно облегчит интеграцию JPA в приложение. Например, размещение аннотации @SpringBootApplication в заголовке приложения указывает Spring на автоматическое сканирование классов и внедрение EntityManager по мере необходимости в зависимости от указанной конфигурации.

Каждое приложение, которое имеет дело с базой данных, должно определить прикладной уровень, единственная цель которого состоит в том, чтобы изолировать постоянный код. Простые приложения могут не требовать всех возможностей JPA, и в некоторых случаях накладные расходы на настройку инфраструктуры могут быть неоправданными. Однако по мере роста приложения структура и инкапсуляция JPA действительно приносят им пользу. Использование JPA делает объектный код простым и обеспечивает традиционную среду для доступа к данным в приложениях Java.

3 Разработка и тестирование

3.1 Безопасность

Безопасность информационных систем на базе интернета является особенно актуальной проблемой. Он сводится к методам и алгоритмам для обеспечения каждого из трех уровней безопасности, которые должна иметь каждая информационная система - аутентификация, авторизация и защита данных. Авторизации и доступа к данным, которые объединены в один полный алгоритм для обеспечения безопасности информационной системы. Для аутентификации пользователя также используются данные из своего цифрового сертификата. Все запросы, отправленные в систему, проходят фильтрацию. Данные защищены с помощью цифровой подписи. Закрытый и открытый ключи авторизованного пользователя хранятся в базе данных. Открытый ключ хранится в незашифрованном виде, а закрытый ключ записывается в базу данных в зашифрованном виде. Симметричный ключ для шифрования и дешифрования закрытого ключа пользователя генерируется с использованием специального алгоритма. Алгоритм может быть реализован в каждом веб-приложении независимо от его конкретного предполагаемого использования. В случае приложения «Comelot», был использован CSRF-Token.

Токен CSRF(Cross Site Request Forgery) - это уникальное, секретное, непредсказуемое значение, которое генерируется серверным приложением и передается клиенту таким образом, что оно включается в последующий HTTP-запрос, сделанный клиентом. Когда более поздний запрос сделан, приложение на стороне сервера проверяет, что запрос включает ожидаемый токен, и отклоняет запрос, если токен отсутствует или недействителен. Токены CSRF могут предотвращать атаки CSRF, не позволяя злоумышленнику создать полностью действительный HTTP-запрос, подходящий для передачи пользователю-жертве. Поскольку злоумышленник не может определить или предсказать значение маркера CSRF пользователя, он не может создать запрос со всеми параметрами, которые необходимы приложению для удовлетворения запроса.

Аутентификация - это проверка ваших учетных данных, таких как имя пользователя или идентификатор пользователя и пароль, для подтверждения вашей личности. Затем система проверяет, используете ли вы свои учетные данные. Будь то в общественных или частных сетях, система аутентифицирует личность пользователя с помощью паролей входа. Обычно аутентификация выполняется с помощью имени пользователя и пароля, хотя есть и другие способы аутентификации.

Факторы аутентификации определяют множество различных элементов, которые система использует для проверки личности перед предоставлением индивидуального доступа к чему-либо. Личность человека может определяться тем, что он знает, и, когда речь заходит о безопасности, должны быть проверены как минимум два или все три фактора аутентификации, чтобы дать

кому-то разрешение на доступ к системе. В зависимости от уровня безопасности факторы аутентификации могут отличаться. В случае с данным проектом была выбрана однофакторная аутентификация. Это самый простой способ аутентификации, при котором требуется пароль для предоставления пользователю доступа к определенной системе, такой как веб-сайт или сеть. Человек может запросить доступ к системе, используя только одну из учетных данных, чтобы подтвердить свою личность. Например, только требование пароля против имени пользователя будет способом проверки учетных данных для входа с использованием однофакторной аутентификации.

Авторизация происходит после того, как ваша личность успешно аутентифицирована системой, что дает вам полный доступ к таким ресурсам, как информация, файлы, базы данных, и т. д. Однако авторизация подтверждает ваши права на предоставление вам доступа к ресурсам только после определения вашей способности доступа система и до какой степени. Другими словами, авторизация - это процесс определения того, имеет ли аутентифицированный пользователь доступ к конкретным ресурсам. Хорошим примером этого является то, что после проверки и подтверждения идентификатора и пароля сотрудника с помощью аутентификации следующим шагом будет определение того, какой сотрудник имеет доступ к какому этажу, и что делается с помощью авторизации.

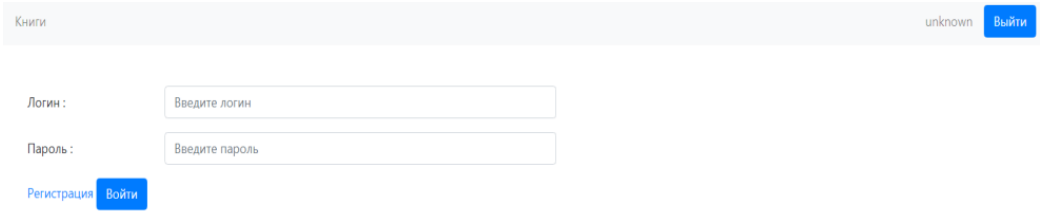

Рисунок 3.1 – Форма авторизации

Доступ к системе защищен аутентификацией и авторизацией, и они часто используются совместно друг с другом. Хотя оба имеют разные концепции, они имеют решающее значение для инфраструктуры веб-служб, особенно когда речь идет о предоставлении доступа к системе. Код авторизации представляет собой набор символов, которые хранятся в памяти системы и позволяют идентифицировать права пользователя. Онлайн авторизация позволяет пользователям использовать сервисы без личного посещения финансовых учреждений, магазинов, учебных заведений. Для этого нужно войти на сайт, перейти по соответствующей ссылке или нажать на определенную кнопку, ввести данные в форму. Лица не авторизованные в базе данных не имеют доступа к библиотеке.

3.2 Описание страниц перехода

Страница входа является первой на пути к пользованию библиотекой, если пользователь ранее не был зарегистрирован, он может пройти к странице регистрации нажав соответствующую кнопку «Регистрация» на странице входа [Приложение А].

Основой приложение принято считать главную страницу. К главной странице имеют доступ пользователи, которые уже числятся в базе данных и прошли авторизацию. После подтверждения своей личности, пользователь имеет право к переходу на страницу книг, где каждый участник может выложить свою книгу в общий доступ [Приложение Б].

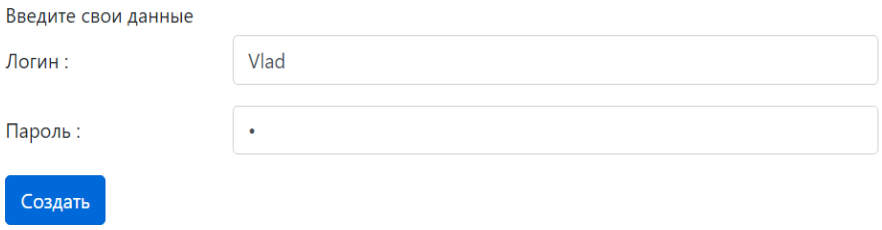

Рисунок 3.2 – Страница регистрации

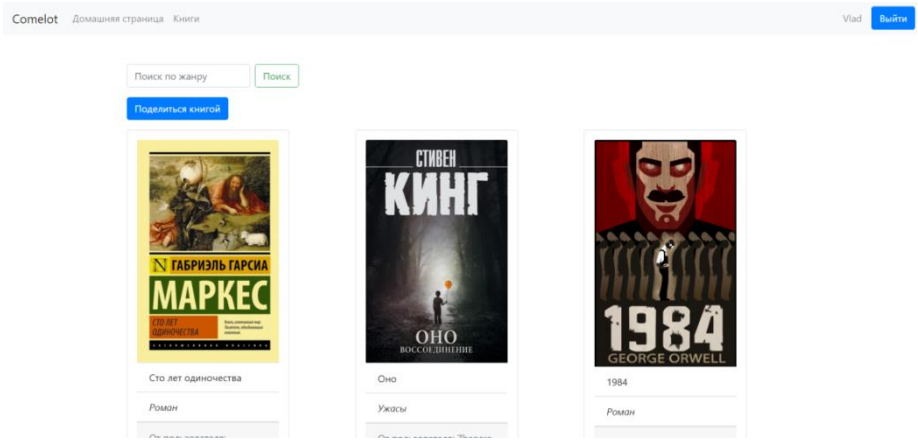

Рисунок 3.3 – Страница со списком книг

Если необходимо провести поиск по определенному жанру, есть возможность перехода к форме поиска, введя необходимый жанр, система будет искать соответствующие запросу, релевантные файлы. Если в графе поиска ввести поиск по жанру «Роман», то перед вами появятся книги по данному жанру, с нижней части формы можно увидеть, кем была размещена книга.

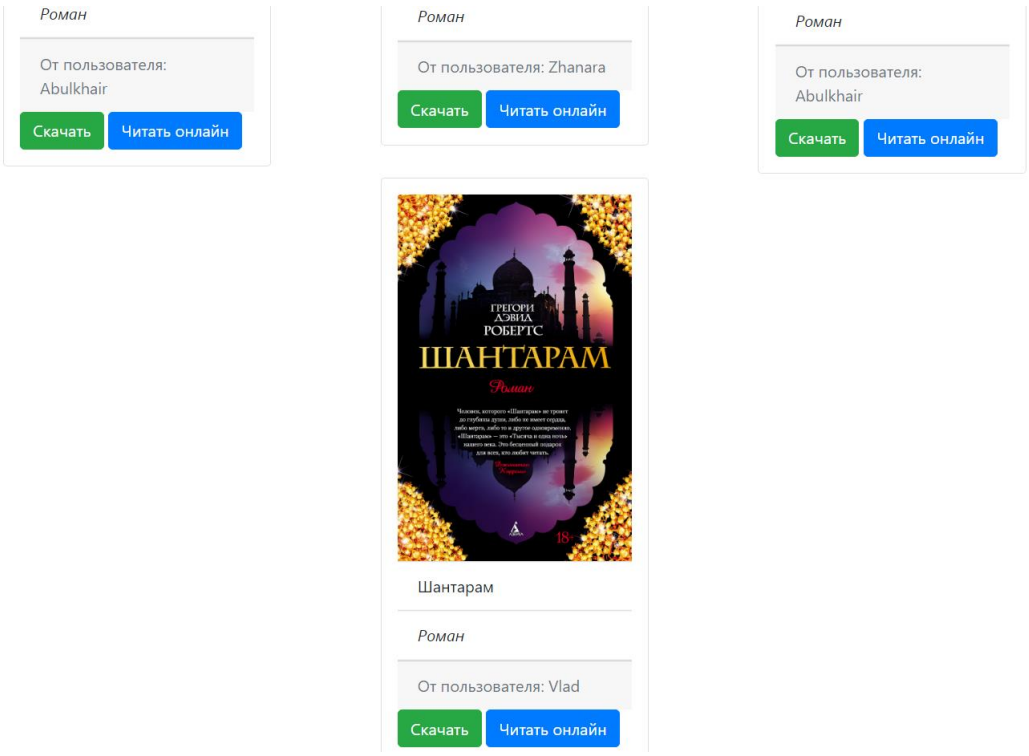

Рисунок 3.4 – Результат поиска по жанру «Роман»

Добавление новых книг осуществляется нажатием по кнопке «Поделиться книгой». Далее идет форма, которую необходимо заполнить, что бы она была добавлена в базу данных.

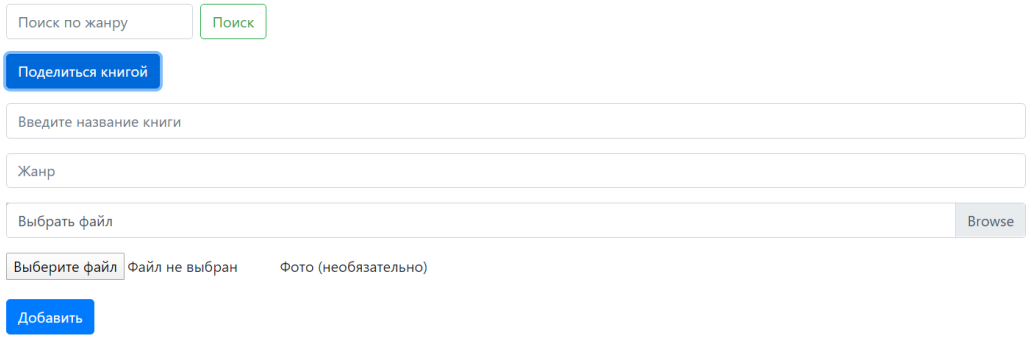

Рисунок 3.5 – Форма добавления книг

После добавления книг, документ добавляется в базу данных и может быть отображен на главной странице. В зависимости от того кто загрузил материал, ниже заголовка будет отображено имя пользователя, загрузившего книгу.

ЗАКЛЮЧЕНИЕ

В эпоху цифровых носителей, людям все легче черпать знания из разных источников, это повышает потребность в онлайн хранилищах и площадках, где можно найти интересующую нас литературу. Данная работа была посвящена анализу уже существующим библиотекам, так же была разработана площадка «Comelot», где пользователи могут размещать собственные книги.

В рамках данного проекта была разработана информационная система. Система, предназначенная для хранения, поиска и обработки информации. В качестве инструментов разработки использовался универсальный фреймворк с открытым исходным кодом для Java-платформы, именуемый Spring. В качестве базы данных был использован PostgresSQL и инструмент pgAdmin, который представляет собой графический клиент для работы с сервером через который мы в удобном виде можем создавать, удалять, изменять базы данных и управлять ими. В качестве сборщика проектов использовался Maven, являющейся подмножеством XML. Был использована спецификация Java EE и Java SE описывающая систему управления сохранением Java-объектов в удобной виде, и использующий сущности (entity). Сущность - это легковесный хранимый объект бизнес логики.

Для обеспечения безопасности были разработаны сервисы аутентификации и авторизации, поддерживаемые CSRF-токеном, для обеспечения утечки пользовательских данных.

В качестве Front-end инструмента использовался Bootstrap, хранящий набор инструментов для создания оформления сайтов и веб-приложений. Который включает в себя и CSS-шаблоны оформления для веб-форм, кнопок, меток и прочих компонентов веб-интерфейса.

Данная работа позволила мне расширить познания в веб-разработке, поскольку был использован один из самых много функциональных и востребованных фреймворков на рынке. Итоговый проект можно считать как начальную часть крупного проекта, который можно запустить на отдельном сервере и при желании улучшить для дальнейшего коммерческого использования.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1 Бауэр К. Java Persistence API и Hibernate [Текст] / Г.Кинг –ДМК Пресс, 2018. – 652 с.

2 Моргунов Е. П. PostgreSQL. Основы языка SQL: учеб. Пособие [Текст] / Е. П. Моргунов; под ред. Е. В. Рогова – БХВ-Петербург, 2018. - 336 с.

3 Уоллс К. Spring в действии [Текст]. / Уоллс К. ДМК Пресс, 2015. – 754 с.

4 Spring Framework Reference Documentation [Электронный ресурс] – Режим доступа: https://spring-projects.ru/projects/spring-framework/.

5 Ликнесс Д. Бессерверные приложения: архитектура, шаблоны и реализация в Azure ,/ Ликнесс Д. - Microsoft Developer Division, 2018 г. – 60 с.

6 Bootstrap Documentation v4.1 [Электронный ресурс] – Режим доступа: https://getbootstrap.com/docs/4.1/getting-started/introduction/.

7 П. Нимейер, Леук Д. Программирование на Java [Текст]./ Леук Д. – Эксмо, 2017 г. - 1216 с.

8 Уэйн К. Алгоритмы на Java. Роберт Седжвик [Текст]. / Седжвик Р – Вильямс, 2019 г. – 848 с.

9 Обзор способов и протоколов аутентификации в веб-приложениях [Электронный ресурс]. / Д. Выростков, тематический блок Habr,2015. –Режим доступа: https://habr.com/ru/company/dataart/blog/262817/.

10 Краткое знакомство с Maven [Электронный ресурс]. / Сергей Штукатуров – портал Tproger, 2019. – Режим доступа: https://tproger.ru/articles/maven-short-intro/.

11 Ю.А.Родичев. Нормативная база и стандарты в области информационной безопасности [Текст]. / Ю. Родичев – Питер,2017. – 256 с.

Приложение А

```
package com.example.comlt.controller;

import com.example.comlt.domain.Role;
import com.example.comlt.domain.User;
import com.example.comlt.repos.UserRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import java.util.Collections;
import java.util.Map;

@Controller
public class RegistrationController {
   @Autowired
   private UserRepo userRepo;
   @GetMapping ("/ registration")
   public String registration () {
      return "registration";
   }

   @PostMapping ("/ registration")
   public String addUser (User user, Map <String, Object> model) {
      User userFromDb = userRepo.findByUsername (user.getUsername ());
      if (userFromDb! = null) {
         model.put ("message", "User exists!");
         return "registration";
      }

      user.setActive (true);
      user.setRoles (Collections.singleton (Role.USER));
      userRepo.save (user);
      return "redirect: / login";
   }
}
```

Приложение Б

```
package com.example.comlt.controller;

import com.example.comlt.domain.Message;
import com.example.comlt.domain.User;
import com.example.comlt.repos.MessageRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;

import java.io.File;
import java.io.IOException;
import java.util.Map;
import java.util.UUID;

@Controller
public class MainController {
    @Autowired
    private MessageRepo messageRepo;
    @Value("${upload.path}")
    private String uploadPath;

    @GetMapping("/")
    public String greeting(Map<String, Object> model) {
        return "greeting";
    }

    @GetMapping("/main")
    public String main(@RequestParam(required = false, defaultValue = "") String
filter, Model model) {
        Iterable<Message> messages = messageRepo.findAll();

        if (filter != null && !filter.isEmpty()) {
            messages = messageRepo.findByTag(filter);
        } else {
            messages = messageRepo.findAll();
        }
```

Продолжение приложения Б

```
model.addAttribute("messages", messages);
        model.addAttribute("filter", filter);

        return "main";
    }


    @PostMapping("/main")
    public String add(
            @AuthenticationPrincipal User user,
            @RequestParam String text,
            @RequestParam String tag, Map<String, Object> model,

            @RequestParam("file") MultipartFile file
    ) throws IOException {
        Message message = new Message(text, tag, user);

        if (file != null && !file.getOriginalFilename().isEmpty()) {
            File uploadDir = new File(uploadPath);
            if (!uploadDir.exists()) {
                uploadDir.mkdir();
            }

            String uuidFile = UUID.randomUUID().toString();
            String resultFilename = uuidFile + "." + file.getOriginalFilename();
            file.transferTo(new File(uploadPath + "/" + resultFilename));
            message.setFilename(resultFilename);
        }

        messageRepo.save(message);
        Iterable<Message> messages = messageRepo.findAll();
        model.put("messages", messages);

        return "main";
    }
}
```