РЕФЕРАТ

Выпускная квалификационная работа содержит 47 страниц, 10 рисунков, и 6 таблиц. Список использованных источников содержит 6 позиций.

Информационная система с Web-интерфейсом для курса дистанционного обучения по дисциплине «Хранение и обработка больших данных»

Выпускная квалификационная работа посвящена вопросу проектирования и реализации системы дистанционного обучения по курсу «хранение и обработка больших данных».

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Компьютеризация всех областей деятельности в наше время продолжается все наращивая свой темп. В связи с прогрессом, ускоряющим свой ход, многие аспекты жизни человека переходят в «облако». Интернет стал неотъемлемой частью нашей жизни буквально за несколько мгновений, по историческим меркам. Торговля, общение, работа и даже образование, постепенно становятся доступны каждому благодаря сети интернет.

Системы дистанционного обучения вероятно один из главных факторов ускорения технического прогресса. Экономия времени, доступность и непрерывность - это малый список того что дают такие системы.

В перспективе, подобные системы ведут к формированию стандартизированной, планетарной системы образования, доступной каждому, что несомненно положительно скажется на человечестве в целом. Но возвращаясь из футуристичных прогнозов к объективной реальности стоит сказать, что уже сейчас, системы дистанционного обучения плотно вошли в обиход людей, особенно в свете последних событий.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. Рассмотрение существующих решений.

История дистанционных курсов началась в 1728 году. Тогда Калеб Филипс подал в бостонскую газету объявление о наборе студентов для изучения стенографии в любые точки страны. Позже, в 1873 году были созданы первые заочные школы США. Вскоре после этого в Чикаго была создана первая программа дистанционного обучения. ([1])

С того времени и до наших дней, дистанционное образование набирало популярность в силу множества преимуществ.

К преимуществам подобного подхода образования, можно отнести следующее:

1. Комфортные условия обучения
2. Более широкий выбор дополнительных курсов
3. Возможности повышения квалификации
4. Отсутствие территориальных и иных требований, характерных для очного образования
5. Разнообразие программ обучения

К недостаткам относят:

1. Проблемы с качеством курсов
2. Проблемы с мотивацией самообучения
3. Качество системы оценивания

Недостатки дистанционного обучения в основном связаны с трудностями перехода от традиционного обучения, и, несомненно решаемы со временем, однако уже сейчас необходимо активно внедрять этот тип обучения в нашу жизнь.

Интернет сам по себе, как средство для обмена информацией, изначально содержал некие образовательные ресурсы, будь то простейшее распространение электронных книг или системы сбора информации.

К наиболее популярной системе сбора информации можно отнести «Wikipedia». Создатели системы «Wikipedia» позиционируют свой проект как свободную энциклопедию. Также помимо «Wikipedia» есть множество сервисов более узкой тематики, однако достоверность информации с этих ресурсов остается под вопросом.

В использовании технологии дистанционного обучения помимо образовательных учреждений заинтересованы также коммерческие предприятия. Примером компании, активно продвигающей дистанционное обучение может служить CISCO, мировой лидер в области сетевых технологий. Фактически система CISCO построена следующим образом - студентам выдается логин и пароль, при помощи которых они входят в систему и получают доступ к лекциям, а также выполнению лабораторных работ, по окончании обучения они сдают экзамен и получают сертификат.

На сегодняшний день существуют готовые комплексные системы для организации учебного процесса, одной из таких систем является Moodle. Данная система распространяется по бесплатной лицензии. К основным плюсам данной системы можно отнести её универсальность и абстрактность архитектуры, создатели хотели разработать общее решение подходящие под все возможные потребности обучающих организаций. Однако такое универсальное данной системы можно рассматривать как недостаток в виду её излишней сложности для нового пользователя.

Помимо подобных систем конструкторов, на сегодняшний день существует также сервисы, предоставляющие доступ к прослушиванию качественных онлайн курсов и коммуникации с преподавателем. В число таких систем, входит Coursera и Udemy

Система Udemy основана в мае 2010 года. Система является американской платформой онлайн обучение предназначенная для профессиональных взрослых и студентов разработчики системы приложили особые усилия для привлечения корпоративных тренеров стремящихся создать курсы для своих предприятий

Coursera является системой подобной Udemy. Она разрабатывалась на базе стэндфордского университета. С самого начала сервис был связан со многими университетами, которые публиковали в системе курсы по различным отраслям знаний. Система устроена следующим образом - слушатели лекций помимо доступа к самим лекциям имеют также возможность коммуницировать с сокурсниками, сдавать тесты, экзамены, непосредственно на сайте.

Озвученные выше системы имеют вид веб-сайтов и веб-приложений доступных из сети интернет. Но также стоит упомянуть системы, благодаря которым имеется возможность проведения вебинаров и конференций. Подобные системы имеют не только обучающие, но также корпоративное применение. Примерами таких систем могут служить - Microsoft Teams, Discord, Zoom.

На данный момент, системы, рассмотренные выше, обеспечивают возможность размещения и ознакомления с учебными материалами, однако в дисциплинах где требуется приобретение практических навыков, таких как написание кода, существует затруднение, вызванное отсутствием возможности, у слушателя курса, получить быстрый отклик, желательно автоматический, на выполненное задание, независимо от доступности преподавателя.

1.2. Обоснование и постановка задачи

Сейчас, в свете пандемии, особенно хорошо видно, что система обучения не рассчитана на полную автоматизацию процесса. Особенно это касается общения с преподавателем, и контроля знаний. Тенденции современного мира диктуют всё большую автоматизацию, и всё меньшее участие человека в рутинных задачах.

В контексте программирования рутинными задачами можно считать проверку кода, написанного студентом, и именно на подобной задаче я сконцентрировался в своей работе.

Перечисленные выше сервисы не имеют функционал автоматической проверки домашнего задания, и как раз это является одним из основных узких мест дистанционного обучения в наши дни.

На сегодняшний день безусловно невозможно полностью отказаться от участия преподавателя в учебном процессе, однако максимальная оптимизация учебного процесса способствует удобству как для студента, так и для преподавателя и помимо всего прочего упрощает доступность образования.

Целью данной работы является реализация информационной системы с Web-интерфейсом для курса дистанционного обучения по дисциплине "Хранение и обработка больших данных"

В рамках работы были сформулированы следующие задачи:

1. Провести анализ предметной области, рассматриваемой дисциплины - “ Хранение обработка больших данных”, и применимость к ней существующих доступных система онлайн обучения.
2. Разработать архитектуру информационной системы с автоматизированной проверкой заданий по дисциплине.
3. Реализовать прототип информационной системы автоматизированной проверкой заданий по анализу больших данных с помощью Apache Spark.

1.3. Обзор платформы для создания системы

Для реализация практической части данной работы был выбран скриптовый язык Python. Последние годы Python становится всё более популярным, в том числе и для Web-разработки. Благодаря таким мощным веб-фреймворкам как Django и Flask, положенных в основу таких

высоконагруженных сервисов и приложений как - Pinterest, Instagram и Dropbox.

Для реализации проекта был выбран Фреймворк Django. Django является крайне технологичным современным фреймворком для web-разработки. Он содержит такие высокоуровневые технологии как:

- Встроенную ORM - эта технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования. ([2])
- Технологию миграций, предназначенную для автоматического применения изменения в модели баз данных.
- Встроенную систему аутентификации пользователя и панель администратора, позволяющие не разрабатывать системы, которые в 90 процентах случаев реализованная одинаково во всех проектах.
- Также Django использует технологию форм, формы в Django это инструмент упрощение автоматизации работы с данными frontend и backend.

Кроме того, фреймворк Django задаёт стандартизированную структуру проекта, благодаря которой гораздо легче расширять свой проект. Django позволяет разделить свою проект на несколько приложений, тем самым еще более ускоряя разработку с использованием готовых решений.

Еще один несомненный плюс Django это встроенная защита от межсетевого скриптинга, подделки межсайтовых запросов, а также от внедрения SQL.

В качестве СУБД мною была выбрана система SQlite, это крайне мощная встраиваемая в приложение база данных, полностью обеспечивающая стандартный функционал CRUD.

Стоит так же упомянуть что в системе так же используется технология «AJAX». В статье «Самый не понимаемый язык программирования в мире стал самым популярным в мире языком программирования» (Crockford, 2008) Дуглас Крокфорд утверждает, что лидирующую позицию «JavaScript» занял в связи с развитием «AJAX».

До ведения технологии «AJAX» приложение работали по классической серверной архитектуре:

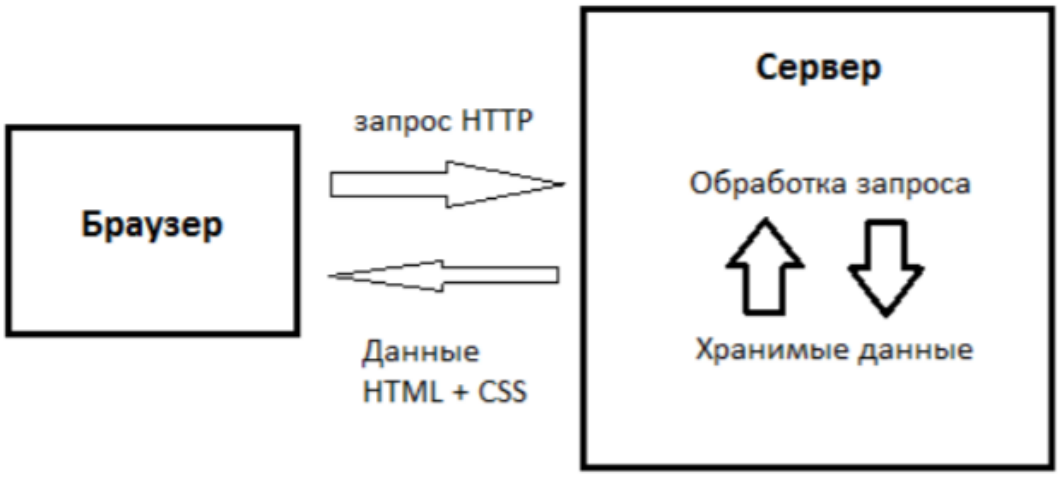


Рис. 2.1.

По мере развития технологии «AJAX», позволяющий осуществить передачу http-запроса без перезагрузки страницы архитектура приложения стала выглядеть следующим образом:

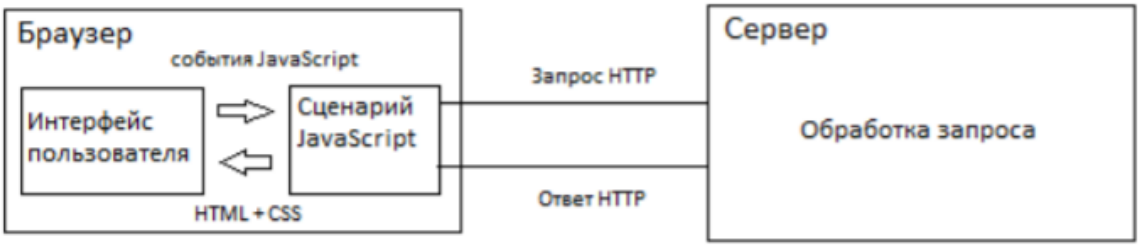


Рис. 2.2.

Применение данного технологического стека довольно стандартно для разработки web-приложений последних лет, поэтому для разработки базового

интерфейса web-сервиса, и основной серверной части, были выбраны озвученные выше технологии.

Но помимо реализации основы web-сервиса, моя работа сосредоточена главным образом на реализации системы автоматической проверки кода студентов, и обеспечения удобного редактирования в браузере.

Для редактирования в реальном времени, подсветки синтаксиса, и поддержки технологии автодополнения, которая призвана упростить и ускорить написание кода, был использован Ace Editor.

Ace Editor — это JavaScript компонент, позволяющий обеспечить требуемые функции. данный редактор кода был выбран по двум причинам –

- во-первых, он поддерживает «из коробки» множество языков программирования, в том числе язык Scala, необходимый для изучения Apache Spark.
- Во-вторых, данный редактор гораздо лучше документирован чем рассмотренные аналоги редакторов кода, доступные на сегодняшний день.

Помимо Ace Editor, были рассмотрены такие редакторы кода как Monaco-Editor и CodeMirror.

CodeMirror был отброшен сразу, в виду того что большая часть функционала работающая в Monaco Editor и Ace Editor из коробки, доступна только с расширениями.

Monaco Editor был отброшен по причине скудного количества языков, для которых поддерживается подсветка синтаксиса.

Исходя из вышеописанных рассуждений, был выбран Ace Editor, как наилучший вариант.

В качестве системы сборки проектов был выбран Apache Maven. Apache Maven — это фреймворк для автоматизации сборки проектов на основе описания их структуры в специальных файлах.

Apache Maven обеспечивает стандартное расположение папок для своих проектов. Когда приходит время создавать новый проект Maven, разработчику приходится вручную создавать каждую папку, что может быстро стать утомительным. И это тот самый момент, когда архетипы Maven приходят на помощь. Архетипы Maven являются предопределенными шаблонами проектов, которые могут быть использованы для создания новых проектов. Проекты, созданные с помощью архетипов, будут содержать все папки и файлы, необходимые для работы. ([3])

Далее, для автоматизации работы в терминале maven, был использован модуль subprocess языка Python.

1.4. Общее техническое задание

Система предназначена для обучения студентов Московского авиационного института. Подразумевается, что систему будут использовать в целях повышения качества и удобства обучения по дисциплине «Хранение обработка больших данных».

Реализация должна представлять собой программный продукт, позволяющий обеспечить студентов лекционным и методическим материалом, а также возможностью автоматической проверки учебных заданий на языке Scala и Java, с использованием фреймворка Apache Spark.

Также для реализации минимально удобной среды для обучения, мною было принято решение реализовать простейший, минимально необходимый функционал для представления лекционного и методического материала к заданию, а также администрирования всего функционала разрабатываемой системы преподавателем.

В связи с этим к системе предъявляются следующие требования:

1.Размещение преподавателем лекционного материала

2.Размещение преподавателем заданий к лекционному материалу

3.Ознакомление студента с лекционным материалом

4.Ознакомление студента с заданием

5.Общение преподавателя со студентом

6.Отправка выполненного задания

7.Автоматический запуск Unit тестов для отправленного студентом задания

Предполагается что тесты написаны преподавателем.

1.4.1. Требования к пользователям системы

Пользователи системы следует разделить на две группы преподаватель-администратор, и студенты.

Слияние роли преподавателя и администратора обусловлено небольшим размером системы. Предполагается излишним расширять систему иными группами пользователей в связи с формальным ограничением системы на одной дисциплине, однако в дальнейшем при развитии системы не представляется сложным дополнить эти группы.

Преподаватель-администратор должен иметь возможность добавления лекционного и методического контента на соответствующие разделы веб-сервиса, а также возможность просмотра логов сдачи практических заданий.

Студенты должны иметь возможность просмотра лекционного материала, просмотра методического материала, а также ввода кода, который требуется написать согласно практическому заданию.

1.4.2. Требования к механизмам взаимодействия с системой

Взаимодействие между пользователями должно осуществляться главным образом путем навигации по разработанным web-страницам системы. также у преподавателя должна быть возможность написать Unit test к составляемому заданию.

У преподавателя и студентов должна быть возможность коммуникации через в систему.

1.4.3. Требования к системе хранения данных и технологиям реализации

Систему хранения данных необходимо сформировать так, чтобы она могла обеспечивать возможность сохранения лекционного материала, методического материала, протоколов сдачи практических заданий и переписки преподавателя со студентами.

Физическую реализацию требуется организовать на ЭВМ с операционной системой Linux.

также рекомендуется поддержка следующих сервисов:

- веб-сервер Nginx
- SQLite

Компьютер должен быть подключён к широкополосному каналу сети интернет.

программную реализацию требуется осуществить на языках HTML, Python, JavaScript, и фреймворке Django.

1.5. Сценарии использования системы

Для более ясных сценариев использования системы, была построена диаграмма вариантов использования. На данной диаграмме можно наглядно наблюдать процессы взаимодействие пользователей с системой.

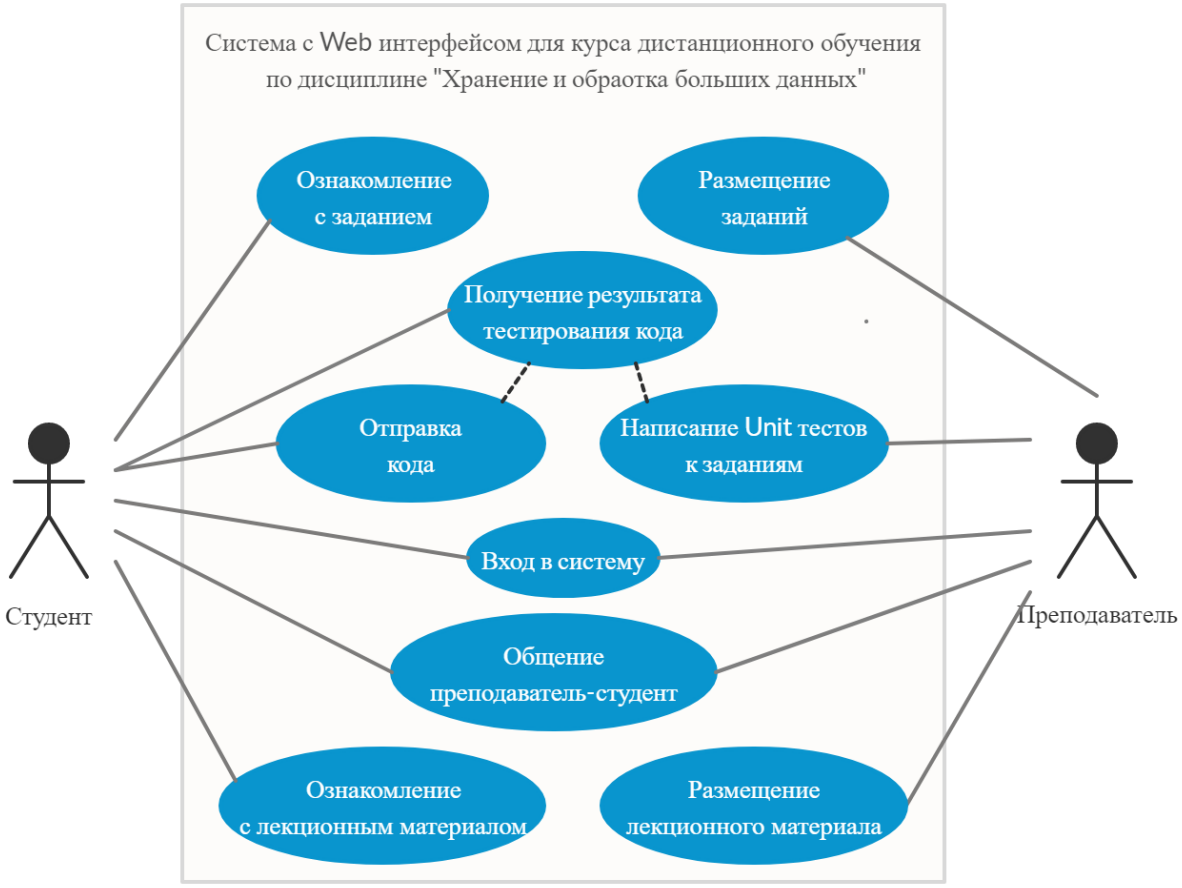


Рис. 2.1. Диаграмма вариантов использования

1.5.1. Сценарий «Размещение заданий»

Цель: Разместить задание и методический материал.

Предусловия: У пользователя есть права администратора

Постусловие: Пользователь завершил редактирование заданий.

Основной поток:

1. Пользователь вводит тему задания.
2. Пользователь редактирует методический материал по средствам markdown разметки.
3. Пользователь добавляет файлы.
4. Пользователь предварительно просматривает конечную структуру страницы и вносит правки, если требуется.
5. Пользователь сохраняет результат и завершает редактирование заданий.

6. Сценарий завершён.

Альтернативные потоки:

А1. Необходимо добавить еще задание.

1. Пользователь сохраняет результат и переходит к следующему заданию на 1 шаг основного сценария.

2. Сценарий завершён.

А2. Необходимо зафиксировать промежуточный результат.

1. Пользователь сохраняет результат и продолжает редактирование.

2. Сценарий завершён.

1.5.2. Сценарий «Размещение лекционного материала»

Цель: Разместить лекционный материал.

Предусловия: У пользователя есть права администратора

Постусловие: Пользователь завершил редактирование лекций.

Основной поток:

1. Пользователь вводит тему лекции.
2. Пользователь редактирует лекционный материал по средствам markdown разметки.
3. Пользователь добавляет файлы.
4. Пользователь предварительно просматривает конечную структуру страницы и вносит правки, если требуется.
5. Пользователь сохраняет результат и завершает редактирование заданий.
6. Сценарий завершён.

Альтернативные потоки:

А1. Необходимо добавить еще лекцию.

1. Пользователь сохраняет результат и переходит к следующему заданию на 1 шаг основного сценария.

2. Сценарий завершён.

А2. Необходимо зафиксировать промежуточный результат.

1. Пользователь сохраняет результат и продолжает редактирование.
2. Сценарий завершён.

1.5.3. Сценарий «Написание Unit тестов к заданиям»

Цель: Написать Unit тесты к заданиям.

Предусловия: У пользователя есть права администратора

Постусловие: Пользователь завершил написание Unit тестов к заданиям.

Основной поток

7. Пользователь редактирует Unit тест к заданию в любой среде разработки или редакторе кода.
8. Сценарий завершён.

1.5.4. Сценарий «Вход в систему»

Цель: Вход в систему.

Предусловия: Пользователь зарегистрирован в системе.

Постусловие: Пользователь вошел в систему.

Основной поток:

1. Пользователь вводит логин и пароль.
2. Пользователь нажимает кнопку «Войти».
3. Сценарий завершён.

Альтернативный поток:

А1. Пользователь не может войти в систему.

1. Пользователь связывается с преподавателем.

2. Сценарий завершён.

1.5.5. Сценарий «Размещение лекционного материала»

Цель: Разместить лекционный материал.

Предусловия: У пользователя есть права администратора

Постусловие: Пользователь завершил редактирование лекций.

Основной поток:

1. Пользователь вводит тему лекции.

2. Пользователь редактирует лекционный материал по средствам markdown разметки.
3. Пользователь добавляет файлы.
4. Пользователь предварительно просматривает конечную структуру страницы и вносит правки, если требуется.
5. Пользователь сохраняет результат и завершает редактирование заданий.
6. Сценарий завершён.

1.5.6. Сценарий «Ознакомление с лекционным материалом»

Цель: Ознакомление с лекционным материалом.

Предусловия: Пользователь зарегистрирован в системе.

Постусловие: Пользователь завершил ознакомление с лекционным материалом.

Основной поток:

9. Пользователь открывает страницу с лекционным материалом.
10. Пользователь ознакомляется с лекционным материалом.
11. Сценарий завершён.

1.5.7. Сценарий «Отправка кода»

Цель: Отправить код на проверку.

Предусловия: Пользователь зарегистрирован в системе.

Постусловие: Пользователь отправил код на проверку.

Основной поток:

1. Пользователь вводит код в редактор кода.
2. Пользователь нажимает кнопку «Отправить».
3. Сценарий завершён.

1.5.8. Сценарий «Ознакомление с заданием»

Цель: Ознакомление с заданием.

Предусловия: Пользователь зарегистрирован в системе.

Постусловие: Пользователь завершил ознакомление с заданием.

Основной поток:

1. Пользователь открывает страницу с заданием.
2. Пользователь ознакомляется с заданием.
3. Сценарий завершён.

1.5.9. Сценарий «Общение преподаватель-студент»

Цель: Отправка сообщения преподавателю.

Предусловия: Пользователь зарегистрирован в системе.

Постусловие: Пользователь отправил сообщение преподавателю.

Основной поток:

1. Пользователь вводит сообщение.
2. Пользователь нажимает кнопку отправить.
3. Сценарий завершён.

1.5.10. Сценарий «Написание Unit теста»

Цель: Написать Unit тест к заданию.

Предусловия: необходимость написать задание.

Постусловие: Преподаватель завершил написание Unit теста.

Основной поток:

1. Преподаватель пишет класс, который необходимо реализовать студенту (необязательно, но желательно).
2. Преподаватель пишет тест к написанному классу.
3. Преподаватель форматирует класс в соответствии с учебной задачей, добавляя разделитель.
4. Сценарий завершён.

Альтернативный поток:

А1. В задании необходимо написать класс целиком.

1. Пользователь сохраняет результат и переходит к следующему заданию на 1 шаг основного сценария.

2. Сценарий завершён.

ПРАКТИЧЕСКАЯ ЧАСТЬ

1.6. Разработка архитектуры системы

1.6.1. Общая архитектура системы

Фреймворк Django предоставляя маршрутизацию запросов, логику отображения, и работу с базой данных посредствам встроенной ORM связывает используемые web-технологии и подсистему тестирования кода, обрабатывая запрос клиентской части, и возвращая ответ.

Подсистема тестирования кода в свою очередь связана с кластером spark.

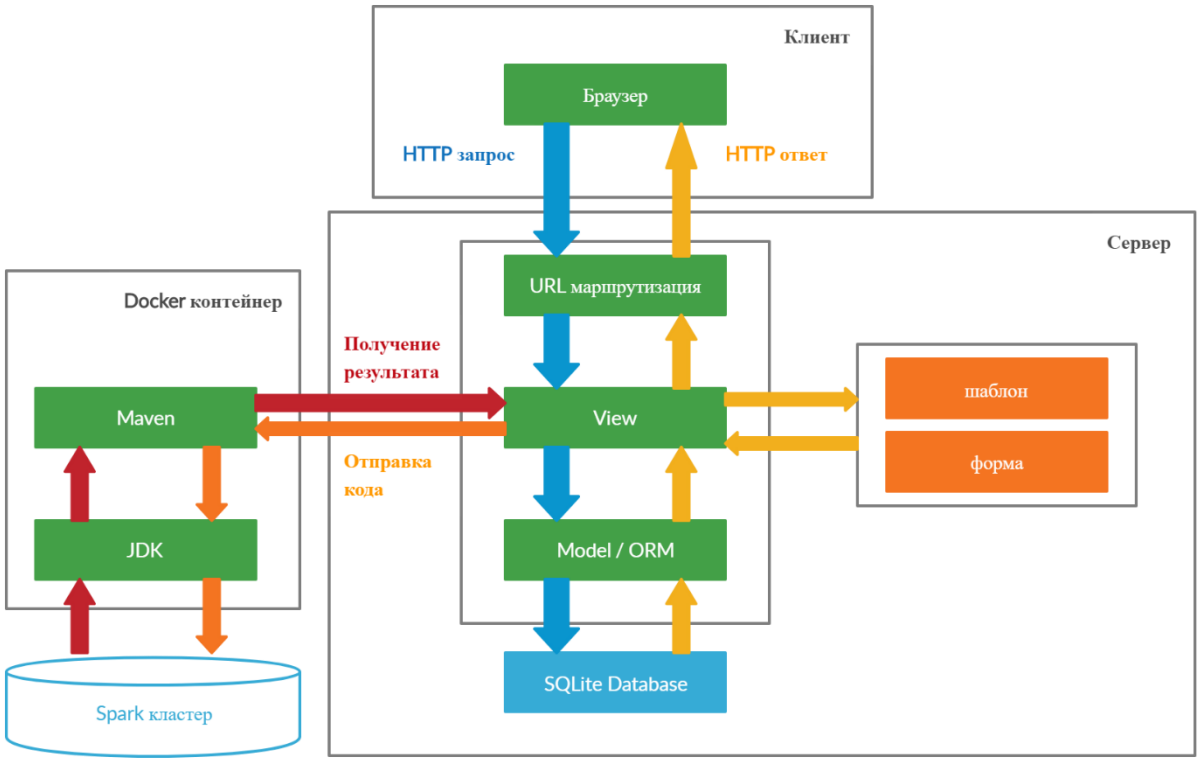


Рис. 2.2.

В проекте Django нужно акцентировать внимание на нескольких файлах:

1. urls.py — файл маршрутизации;
2. views.py — файл контроллера и выдачи рендера страница;
3. models.py — файл работы с базой данных на основе ORM.

Они, как правило, должны быть в каждом приложении, за счет чего реализуется обмен и обработка данными по принципу MTV. Каждый из этих

файлов выполняет свою роль в обработке по принципу MTV, который наследует свою идею от MVC.

Концептуальный принцип MVC реализован из таких частей:

- *M* — доступ к данным, обрабатывается слоем работы с базой данных.
- *V* — эта часть, которая определяет какие данные получать и как их отображать, обрабатывается представлениями и шаблонами.
- *C* — эта часть, которая выбирает представление в зависимости от пользовательского ввода, обрабатывается самой средой разработки, следуя созданной схемой URL, и вызывает соответствующую функцию Python для указанного URL.

Но в Django это работает чуть иначе. Так как «C» обрабатывается средой разработки и всё интересное в Django происходит в моделях, шаблонах и представлениях, на Django ссылаются как на MTV-ориентированную среду разработки. В MTV-подходе к разработке:

- *M* — определено для «Модели» (Model), слоя доступа к данным. Этот слой знает всё о данных: как получить к ним доступ, как проверить их, как с ними работать и как данные связаны между собой.
- *T* — определено для «Шаблона» (Template), слоя представления данных. Этот слой принимает решения относительно представления данных: как и что должно отображаться на странице или в другом типе документа.
- *V* — определено для «Представления» (View), слоя бизнес-логики. Этот слой содержит логику, как получать доступ к моделям и применять соответствующий шаблон. Вы можете рассматривать его как мост между моделями и шаблонами.

Ниже представлена схема того, как Django представляет для себя MVC в виде MTV

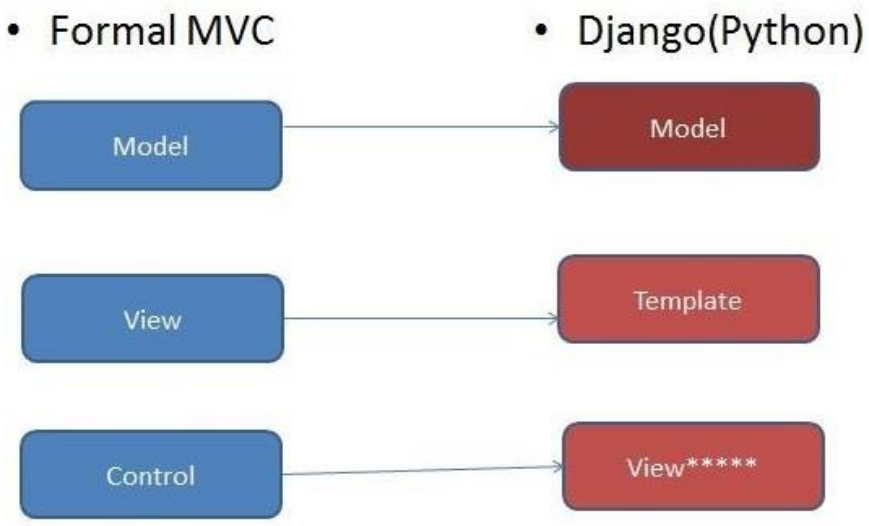
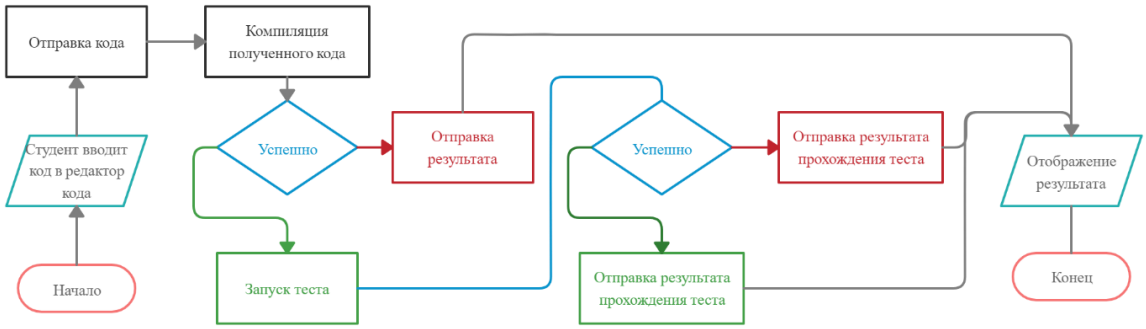


Рис. 2.3. Реализация MTV в Django ([4])

1.6.2. Архитектура системы автопроверки заданий

Для разработки системы автоматической проверки кода, необходимо воспользоваться целым набором различных технологий. Если рассматривать систему абстрактно она выглядит следующим образом:



Студент, взаимодействуя с редактором кода, пишет некоторую функцию на языке Java или Scala. Далее по окончании написания функции, он нажимает на кнопку «отправить». При нажатии кнопки «отправить» код передаётся в .jar файл на сервере со сборщиком maven, при этом этот сервер может разделен с сервером, на котором расположен сайт. После записи кода в файл происходит его компиляция, и формируется скомпилированный класс. Если компиляция вернула ошибку, то пользователь видит сообщение об ошибке, иначе если компиляция успешна, запускаются тесты, написанные преподавателем.

Результат тестов также возвращается пользователю, тем самым реализуя автоматическую обратную связь во время учебного процесса, без непосредственного присутствия преподавателя.

Рассматривая весь процесс более подробно, стоит сказать, что в момент нажатие кнопки «отправить», фактически, создается новый файл на сервере со сборщиком maven, после чего, при помощи модуля subprocess, языка Python, отправляется запрос в терминал сервера сборки проекта maven. Далее это же модуль возвращает ответ командной строки и, если в результате сборки проекта не возникло ошибок, запускается класс с Unit test, написанный ранее преподавателем. а так как запуск unit тестов происходит через терминал при помощи всё того же модуля subprocess, пользователю возвращается результат работа теста.

Отдельно стоит остановится на том, что помимо тестирования, и его результата, студент получает результат преобразований на данных из кластера spark.

В распределенном режиме Spark использует архитектуру, ведущий/ ведомый (master/slave) с одним центральным координатором и множеством распределенных рабочих узлов. Центральный координатор называется драйвером (driver). Драйвер взаимодействует с (возможно) большим числом рабочих узлов, которые называют исполнителями (executors). Драйвер и исполнители выполняются в отдельных и независимых друг от друга процессах Java и составляют приложение Spark.

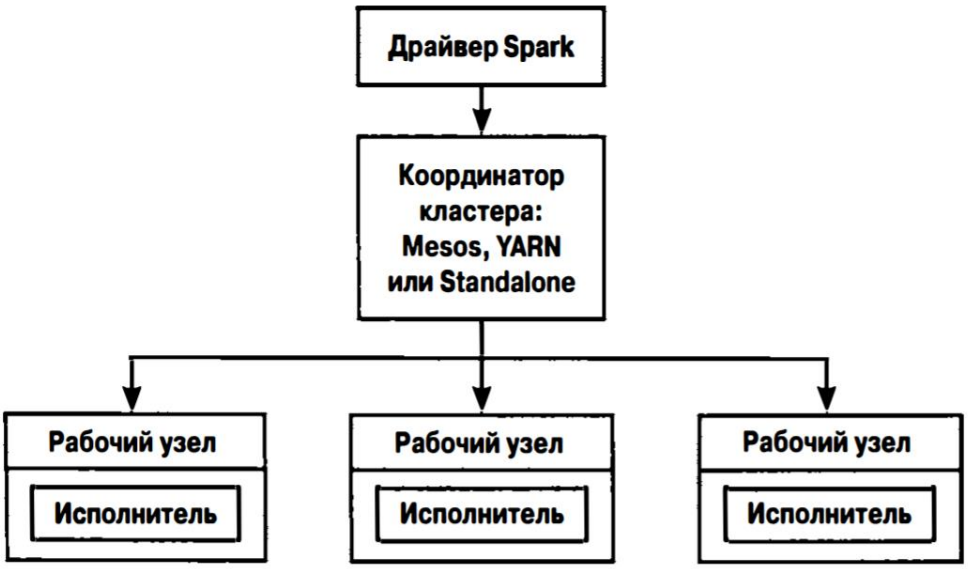


Рис. 2.4.

Драйвер - это процесс, в котором выполняется метод main() программы. Этот процесс выполняет пользовательский код, создающий объект SparkContext, наборы RDD и выполняющий преобразования и действия. Запуская интерактивную оболочку Spark, вы создаете программу-драйвер. С завершением работы драйвера завершается и выполнение приложения. ([5])

В процессе выполнения драйвер решает две задачи:

1. Преобразует пользовательскую программу в задания.
2. Планирует выполнение заданий исполнителями.

Исполнители в Spark - это рабочие процессы, ответственные за выполнение отдельных заданий. Исполнители запускаются один раз и продолжают работать в течение всего жизненного цикла приложения.

Исполнители решают следующие задачи:

- Во-первых, они выполняют задания, переданные приложением, и возвращают результаты драйверу.
- Во-вторых, обеспечивают сохранение в памяти наборов RDD, кэшированных пользовательскими программами, через службу Block Manager, действующую внутри каждого исполнителя.

Диспетчер кластера - это подключаемый компонент Spark. Фреймворк

Spark поддерживает возможность работы поверх внешних диспетчеров, таких как YARN и Mesos, а также поверх встроенного диспетчера Spark Standalone.

В контексте предметной области данной работы рассматривается YARN как диспетчер кластера использующиеся в обучении на данный момент.

Написанный код передается в рамках запускаемого сценария spark-submit, являющегося единым для всех диспетчеров кластера. В рамках этого сценария указываются более тонкие настройки, такие как указания диспетчера класса, пути к файлу с функцией main(), имя приложения, а также регулируется расход памяти. Все это указывается по средствам добавления флагов сценария.

Выше рассмотрен процесс работы системы с точки зрения пользователя. С точки зрения системы стоит пояснить некоторые архитектурные решения, принятые для обеспечения корректной работы системы. Для начала стоит пояснить что система, несмотря на то что оптимизирует процесс обучения, упрощая процесс как мониторинга знаний, так и процесс взаимодействия преподавателя со студентами, всё ещё в достаточной мере зависит от преподавателя, пусть и дает гораздо большую свободу в процессе обучения.

Для написания юнит тестов, применяющихся к коду студентов, необходима подготовка, проводимым преподавателем, которая заключающаяся в написании тестов.

Этот аспект системы устроен следующим образом - преподаватель пишет класс и тест к нему, проверяя его работу, далее преподаватель удаляет либо фрагмент написанного класса, который студент должен написать сам в рамках учебного процесса, либо весь класс, тем самым от студента требуется написание всего класса целиком.

Ниже представлена диаграмма, иллюстрирующая данный процесс:

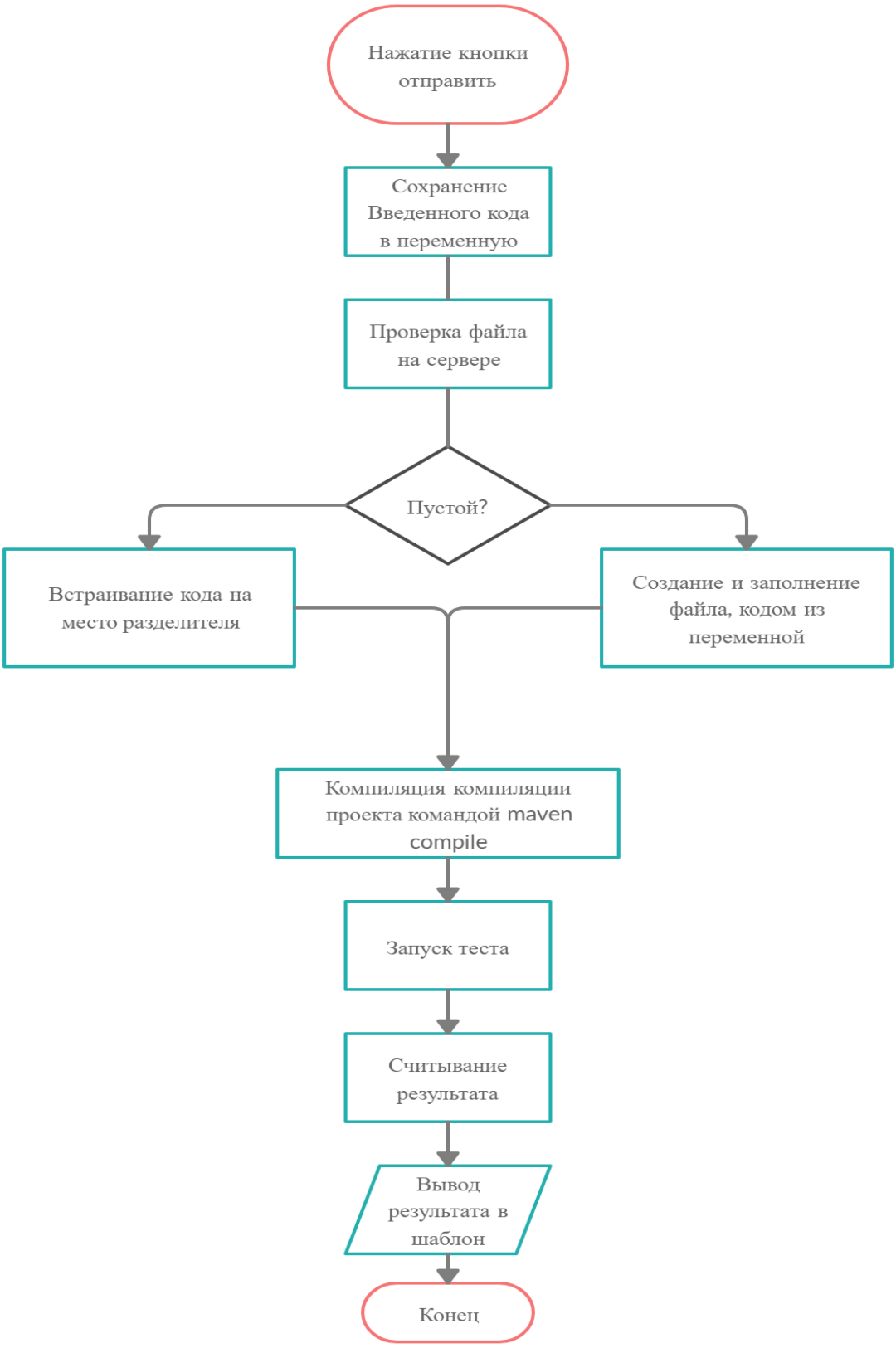

Нажатие кнопки отправить
Сохранение Введенного кода в переменную
Проверка файла на сервере
Пустой?
Встраивание кода на место разделителя
Создание и заполнение файла, кодом из переменной
Компиляция компиляции проекта командой maven compile
Запуск теста
Считывание результата
Вывод результата в шаблон
Конец

Рис. 2.5.

1.7. Разработка интерфейса системы

Одним из главных элементов любой системы основанной на Web-технологиях является интерфейс. Именно с интерфейсом пользователя контактируют всё время пользования сервисом, настоящее время сложно переоценить влияние продуманного и ненагруженного интерфейса на восприятие всего сервиса в целом.

Программный компонент системы, разрабатываемые в этом проекте представляет собой сайт имеющие графический интерфейс. При построении визуальных интерфейсов широко используется понятие «юзабилити», обозначающие итоговый уровень удобство интерфейса для пользователя.

Также со временем стало очевидно, что для успешной и удобной работы с сайтом необходимо следующие компоненты:

- *Блок идентификации сайта*, который чаще всего представляют собой логотип, либо знак, позволяющий точно идентифицировать сайт с первого взгляда.
- Ни один сайт наше время не обходятся без *гиперссылок*, гиперссылка является частью документов, которая ссылается на другой элемент в самом документе.
- Ещё одним неотъемлемым элементом, со временем *стала панель навигации*, несмотря на устоявшееся классическое положение панели навигации в верхнем левом углу, в настоящий момент, панель навигации может довольно широко варьировать свое местоположение.
- Ну и самая главная часть интерфейса сайта — это *блоки с контентом*, являющаяся собственно самим содержанием сайта, представляет из себя совокупность мультимедиа и текстовой информации.

Используя эти понятия можно описать большую часть современных сайтов в сети интернет. Дизайнеры интерфейсов на протяжении многих лет

отслеживали тенденции предпочтений людей в этой области, с течением времени сформировав некий общий шаблон интерфейса. Именно из общепринятых принципов я отталкивался при разработке интерфейса.

Разработанный в системе блок идентификации сайта представляет собой логотип, являющаяся векторным svg файлом. Использование svg файлов оправдано тем, что незначительно уменьшают объем загружаемых пользователем информации, при загрузке страниц.

Также было принято решение расположить элементы меню сверху на прозрачном «navbar», Данное решение было принято по трем основным причинам:

1. элемента интерфейса сверху интуитивно понятным для пользователя,
2. данное решение обеспечивает симметричность расположение элементов интерфейса при адаптации к различным разрешением экрану
3. это позволяет сделать навигационное меню низким по высоте, и растянутым на весь экран, Тем самым обеспечивая удобство просмотра контента при прилипании меню к верхней части экрана во время прокрутки страницы.

Сам веб-сервис представляет из себя совокупность всех приложений с помощью которых осуществляется Размещение отображения электронного материала, я размещение отображение методического материала, авторизация пользователя.

Также не стоит забывать о том, что центральный функционал данного web-сервиса — это автоматическая проверка кода, а значит сервис предоставляет удобный способ редактирования кода отвечающей современным стандартам редакторов кода, и систему автоматической проверки кода.

Высокоуровневость Django позволяет не углубляться в механизм работы сессий, сосредоточившись на разработке визуального интерфейса и серверной логики. интерфейс в приложении, как и во всех подобных в

приложениях, формируется при помощи HTML разметки и CSS стилей. Фреймворк позволяет оптимизировать процесс построения интерфейса при помощи встроенного шаблонизатора реализуя принцип DRY.

К системе есть общий шаблон описывающие внешний вид навигационного меню и футера сайта. далее при помощи механизма шаблонизации, в зависимости от навигации по сайту, в основной шаблон встраивается контент, в сущности представляющий из себя HTML блок, расположенный в другом файле. Данный подход позволяет оптимизировать и унифицировать структуру сайта, минимизируя либо вообще устраняя копирайтинг одного и того же кода.

Кроме механизма шаблонизации и стандартных средств разметки, в отображение интерфейса участвуют механизм Django forms, а также используется свободный набор инструментов Bootstrap.

Ниже приведена карта сайта:

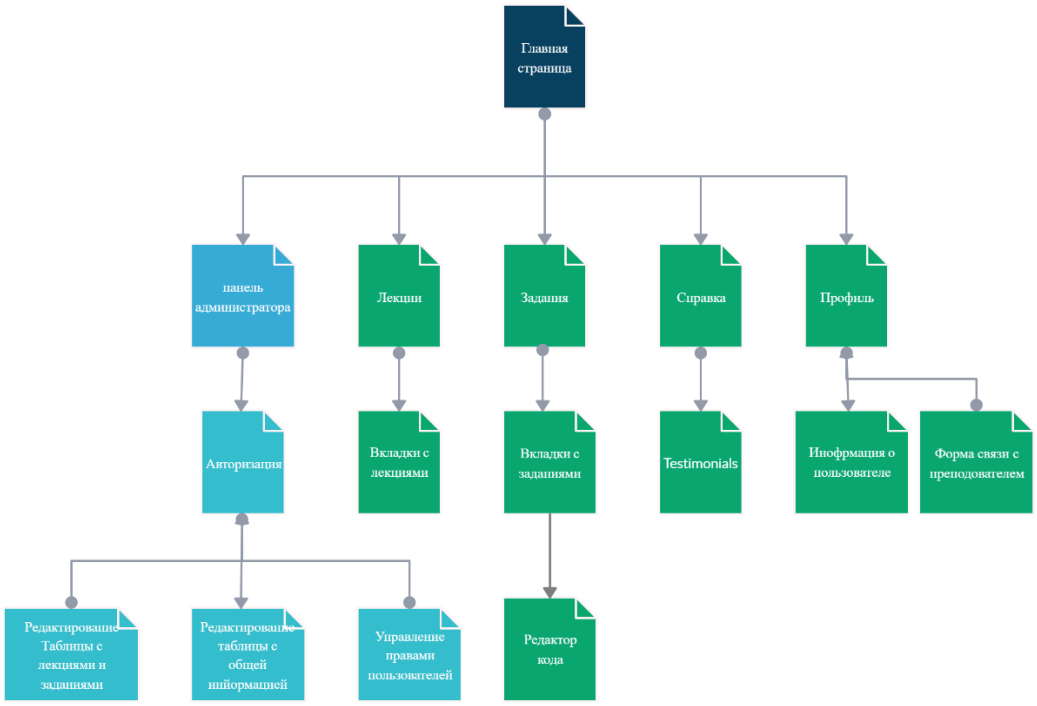


Рис. 2.6.

Из карты ясно видно, что при заходе на сайт, пользователь видит главную страницу, с которой при помощи технологии гиперссылок может перейти в раздел с лекциями, заданиями, справкой по сайту, а также

Посмотреть информацию по профилю. Также на сайте реализована удобная панель администратора, которая благодаря Фреймворку Django имеет стандартизированный внешний вид, что улучшает восприятие пользователям.

Панель администратора позволяет осуществлять редактирование раздела лекций, Раздела заданий, а также добавление и управление группами пользователей, и их правами. Кроме того, панель администратора позволяет просматривать «логи» сообщений, которые отправляют студенты, и протоколы, отображающие историю отправленного на проверку кода.

Главная страница сайта выглядит следующим образом:

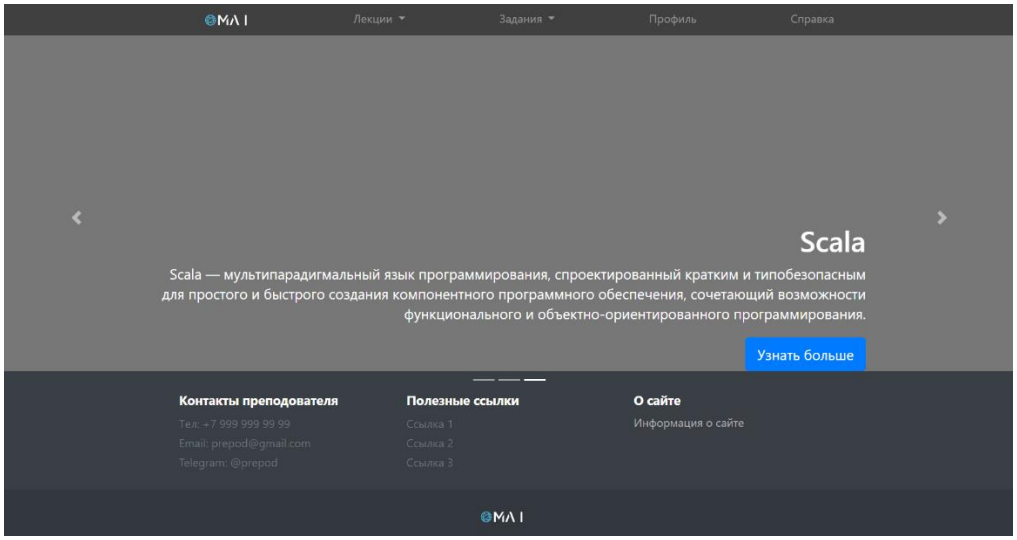


Рис. 2.7.

При заходе на сайт, пользователь видит меню с разделами сайта, настраиваемую информацию, расположенную на футере, а также настраиваемую информацию на элементе отображения типа Carousel.

1.8. Разработка базы данных системы

Для хранения информации в системе дистанционного обучения по курсу «Хранение обработка больших данных» используется СУБД SQLite. Требуется разработать структуру базы данных обеспечивающую продуманную и надежную логистику данных в рамках системы. База данных будет иметь имя «db», и состоять из 6 таблиц. Структура базы данных приведена на рисунке 2.4.

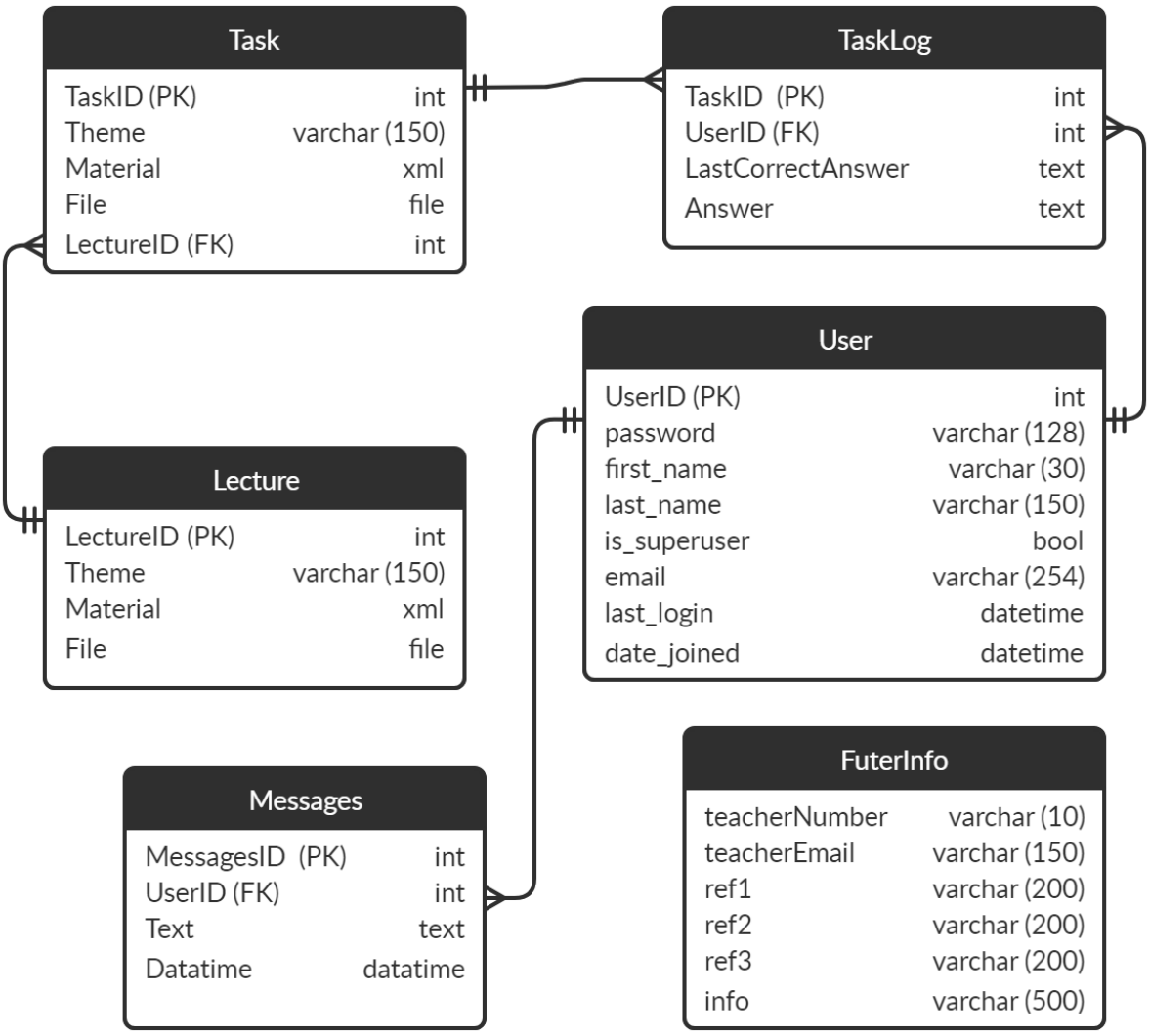


Рис. 2.8. Физическая схема базы данных

Необходимо отметить, что материал в таблице Task и Lecture содержится в полях, хранящих XML файл, благодаря этому возможно обеспечить хранение изображений, текста, а также ссылок с последующей отрисовкой видео плеера в одном поле таблицы. Обе таблицы предоставляют возможность загрузки файлов, которые возможно пригодятся в учебном процессе.

Структура таблиц была создана средствами Django, а именно встроенной ORM системой. Ниже приведен пример моделей Task и Lecture, описанных в виде класса языка python.

```
class Task(models.Model):
    title = models.CharField("Тема", max_length=150)
    TaskBox = MartorField()
    files = models.FileField("Файл", blank=True)

    def __str__(self):
        return self.title

    class Meta:
        verbose_name = "Задание"
        verbose_name_plural = "Задания"

class Lecture(models.Model):
    title = models.CharField("Тема", max_length=150)
    LectureBox = MartorField()
    files = models.FileField("Файл", blank=True)

    def __str__(self):
        return self.title

    class Meta:
        verbose_name = "Лекция"
        verbose_name_plural = "Лекции"
```

Подробное описание базы данных приведено в таблицах 2.1, 2.2, 2.3, 2.4, 2.5, 2.6.

Таблица 2.1.

Название таблицы	Описание таблицы	Название полей	Описание полей
Task	Хранит в себе методический материал, заголовок и прикрепленные файлы. Методический материал находится в виде XML.	TaskID (PK)	Первичный ключ, задается автоматически, имеет тип int.
		Theme	Тема, максимальное число символов - 150
		Material	Материал, представляет из себя XML файл, содержащий мультимедиа, а так же ссылки, редактируется в панели администратора
		File	Поле предназначенное для хранения файла
		LectureID (FK)	Вторичный ключ, для соединения с таблицей Lecture.

Таблица 2.2.

Название таблицы	Описание таблицы	Название полей	Описание полей
Lecture	Хранит в себе лекционный материал, заголовок и прикрепленные файлы. Методический материал находится в виде XML.	LectureID (PK)	Первичный ключ, задается автоматически, имеет тип int.
		Theme	Тема, максимальное число символов - 150
		Material	Материал, представляет из себя XML файл, содержащий мультимедиа, а так же ссылки, редактируется в панели администратора
		File	Поле предназначенное для хранения файла

Таблица 2.3.

Название таблицы	Описание таблицы	Название полей	Описание полей
TaskLog	Хранит в себе код, получаемый при нажатии кнопки «отпрпавить»	TaskID (PK)	Первичный ключ, задается автоматически, имеет тип int.
		UserID (FK)	Вторичный ключ, для соединения с таблицей User.
		LastCorrectAnswer	Последний правильный ответ.
		Answer	Ответ.

Таблица 2.4.

Название таблицы	Описание таблицы	Название полей	Описание полей
Messages	Хранит в себе сообщения оправленные пользователями.	MessagesID (PK)	Первичный ключ, задается автоматически, имеет тип int.
		UserID (FK)	Вторичный ключ, для соединения с таблицей User.
		Text	Текст сообщения.
		Datatime	Дата отправки.

Таблица 2.5.

Название таблицы	Описание таблицы	Название полей	Описание полей
User	Хранит в себе данные пользлвателя.	UserID (PK)	Первичный ключ, задается автоматически, имеет тип int.
		password	Пароль.
		first_name	Имя.
		last_name	Фамилия.
		is_superuser	Администратор или нет.
		email	Почта.
		last_login	Последний вход.
		date_joined	Дата регистрации.

Таблица 2.6.

Название таблицы	Описание таблицы	Название полей	Описание полей
FuterInfo	Хранит в себе общие данные с главной страницы.	teacherNumber	Телефон преподавателя.
		teacherEmail	Почта преподавателя.
		ref1	Ссылка №1.
		ref2	Ссылка №
		ref3	Ссылка №
		info	Общая информация

1.1. Реализация системы автоматической проверки заданий

Для реализации данной системы был использован в сборщик проектов maven, а также модуль subprocess языка Python. Кроме того, для ввода кода, был добавлен Java компонент – Ace Editor.

При подключении Ace Editor был создана следующая форма:

```
24: from django_ace import AceWidget
25: class EditorForm(forms.ModelForm):
26:     class Meta:
27:         model = TaskLog
28:         widgets = {
29:             "Answer": AceWidget(mode='scala', theme='twi
    light',wordwrap=True),
30:         }
31:         exclude = ()
```

Данная форма обеспечивает отображение виджета редактора, который позволяет подсвечивать синтаксис и автоматически дополнять код.

Помимо формы, так же создается и ORM модель, описывающая таблицу где, хранится код.

```
32: class TaskLog(models.Model):
33:     Ansver = models.TextField()
34: 
35:     LustCorrectAnswer = models.TextField()
36:     created_at = models.DateTimeField(auto_now_add=True)
37: 
38:     class Meta:
39:         ordering = ('-created_at', )
```

Суть системы такова - после ввода кода в форму редактора, пользователь нажимает кнопку «отправить», после нажатия кнопки код сохраняется в базу данных, а также при помощи модуля subprocess, передается в терминал и сохраняется в файл с расширением класса данного языка.

После, в терминале вызывается команда:

```
$ mvn compile
```

Передающаяся в терминал при помощи все того же модуля subprocess.

Далее вызывается сценарий spark-submit, в который передается собранный файл и команда:

```
$ mvn exec:java -Dexec.mainClass="Classname”
```

Запускающая скомпилированный класс, и выводящая результат в консоль. Модуль subprocess автоматически считывает результат выполнения команд, и сохраняет их в переменную, которая потом сохраняется в базу данных.

Далее вызывается сценарий spark-submit, в который передается собранный файл.

1.2. Функционал студента

Как было указано выше, студент может осуществлять навигацию по разделам сайта, общаться с преподавателем и отправлять свой код.

Раздел лекций выглядит следующим образом:

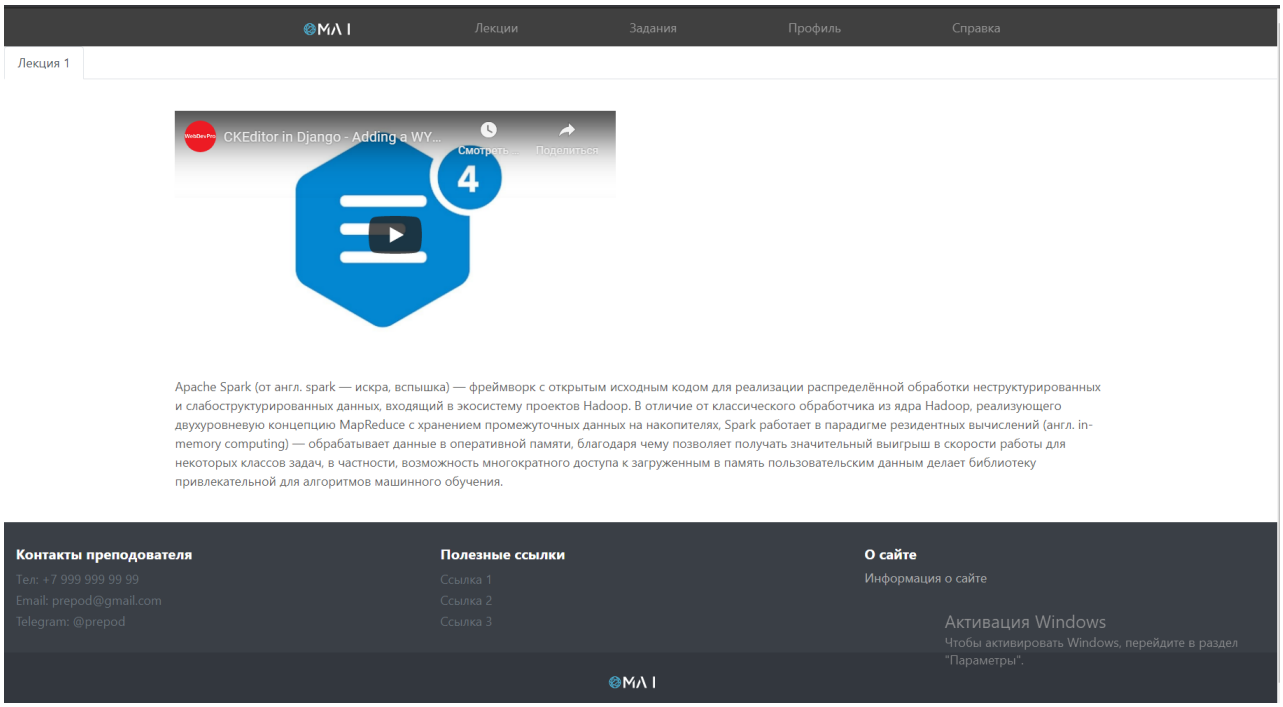


Рис. 2.9.

Структура разделов с лекциями такова - каждая лекция находится на вкладке, это позволяет:

1. удобно осуществлять навигацию по лекционному материалу
2. быстро переходить к необходимой лекции
3. подобная структура обеспечивает хорошую масштабируемость для мобильных устройств.

Кроме того, стоит отметить что лекционный материал размещается в произвольном порядке. В зависимости от нужд преподавателя точка текст, изображение и видео, опционально могут размещаться в любой последовательности, необходимой для ля лучшего восприятия материала.

Далее, студенту доступен раздел «Задания», а котором предполагается размещение методического материала.

Кроме того, в данном разделе присутствует окно редактора кода, для выполнения студентом практических заданий.

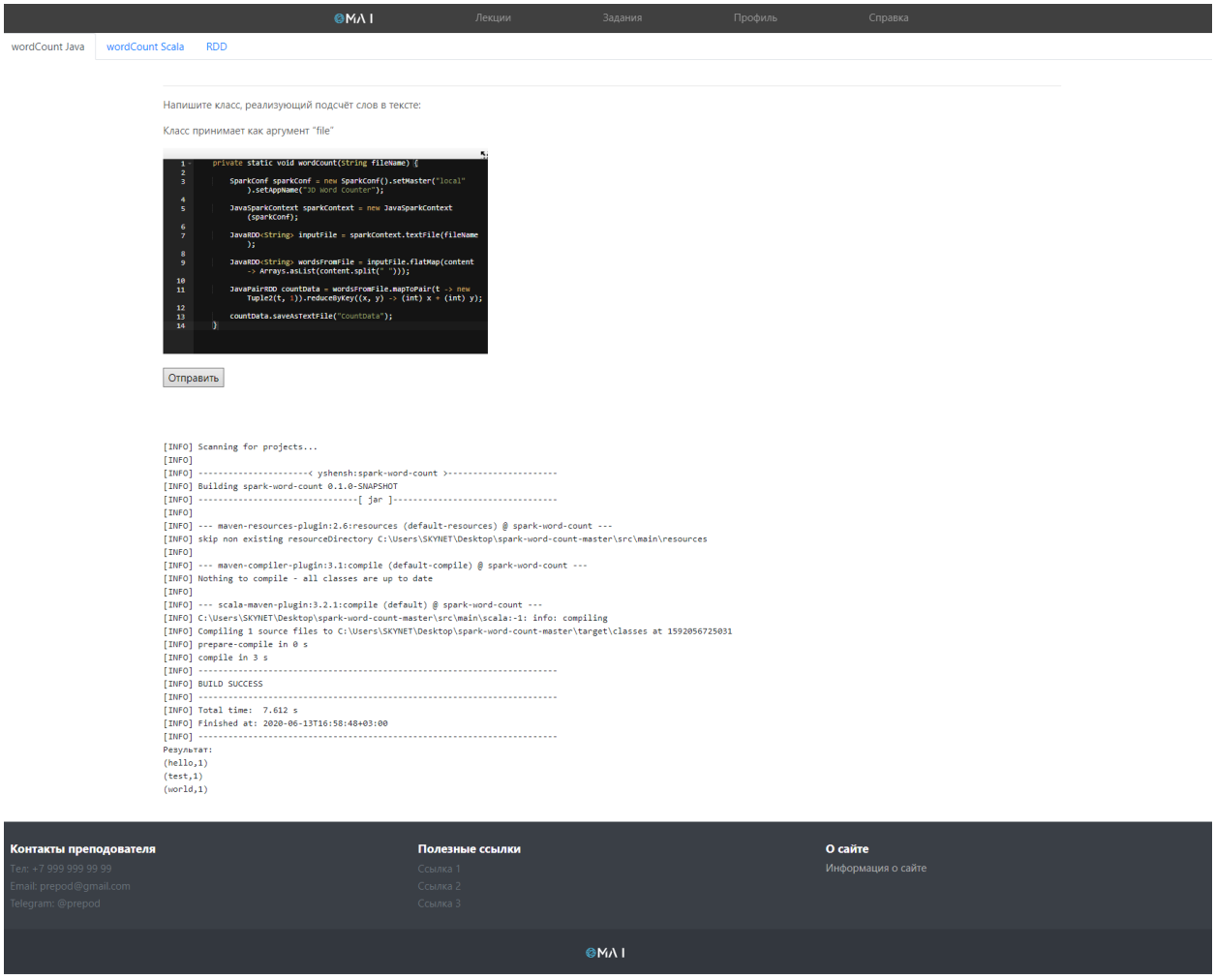


Рис. 2.10.

Редактор кода поддерживает подсветку синтаксиса и автодополнение кода, для отправки кода на проверку, студенту необходимо нажать кнопку «Отправить».

Так же студенту доступна информация о его профиле, в разделе «Профиль»

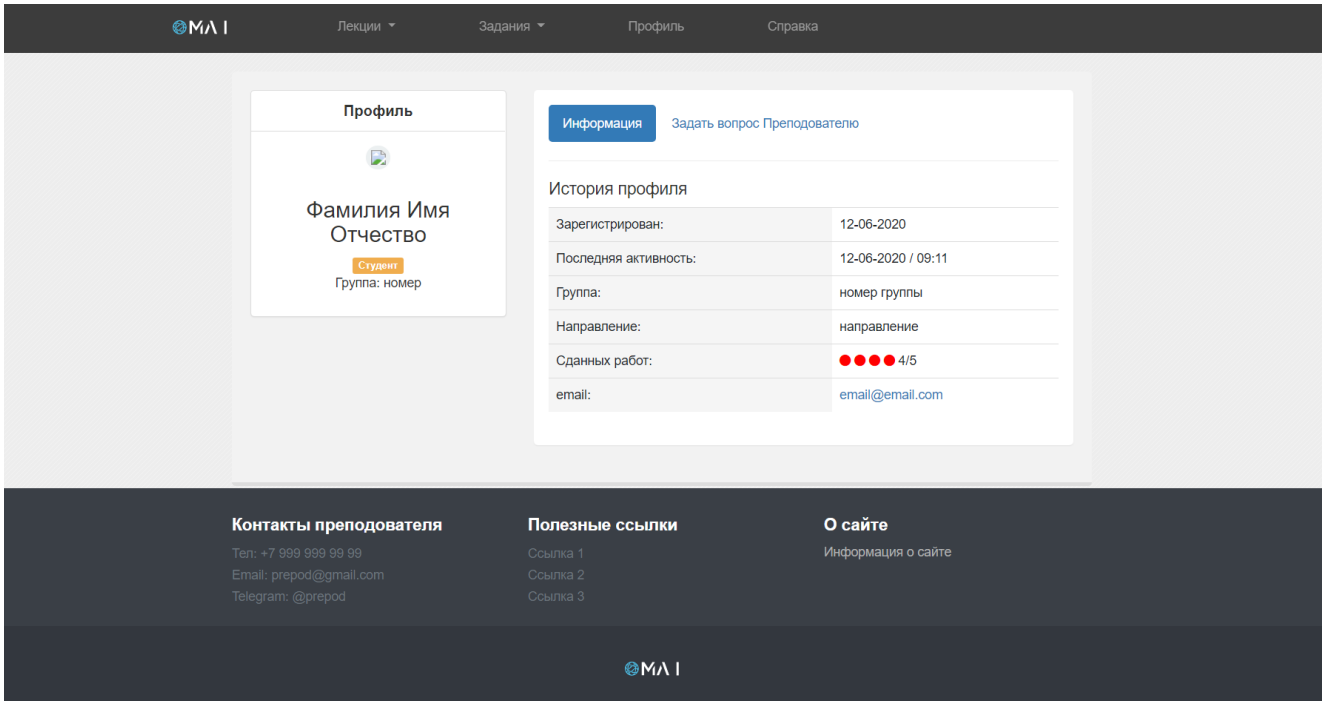


Рис. 2.11.

В данном разделе студент может ознакомится с информацией о сданных работах и написать вопрос преподавателю.

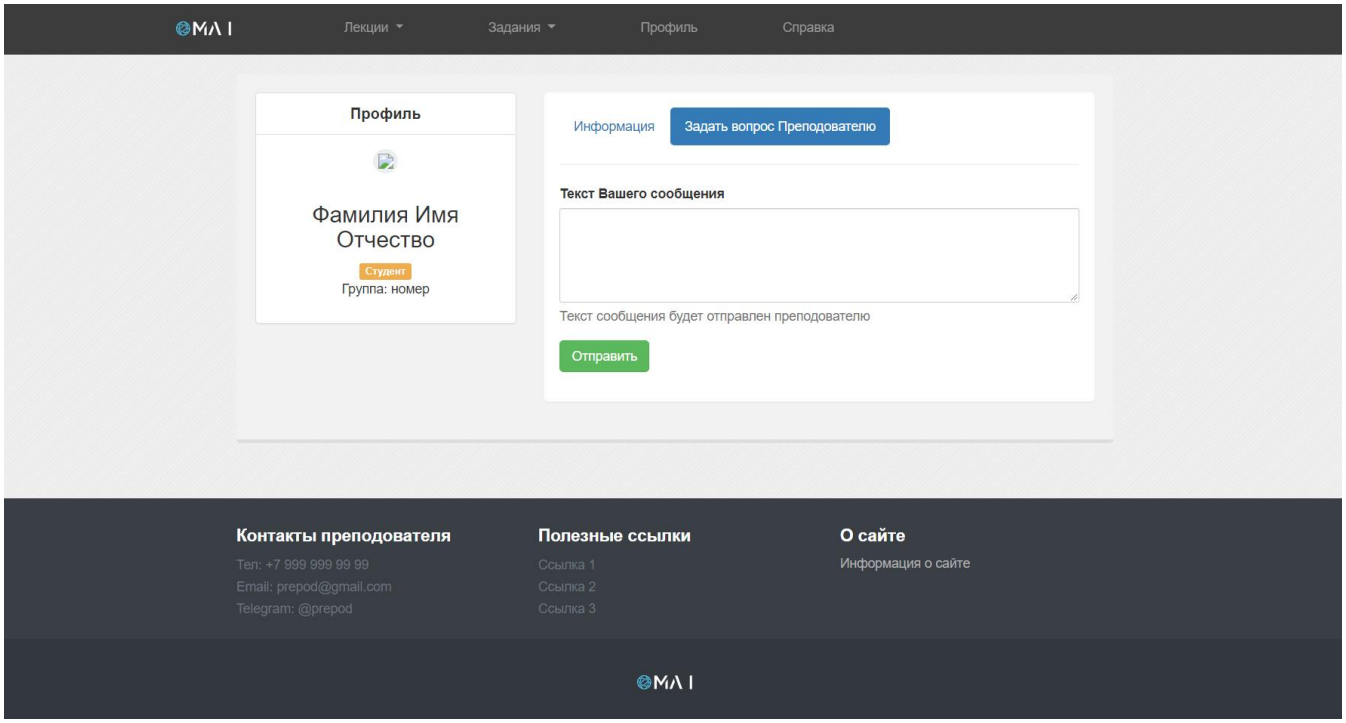


Рис. 2.12.

На данной форме студенту предоставляется интерфейс для написания сообщения преподавателю, и его отправке.

1.3. Функционал преподавателя

Функционал преподавателя представлен теми же возможностями что есть у студента, однако к ним добавляется доступ к панели администратора сайтам. В панели администратора преподаватель может редактировать количество и содержание лекционного материала, количества и содержания заданий, а также просматривать логи сообщений студентов и логи сдачи заданий.

На рисунке ниже изображена окно Редактирование лекционного материала:

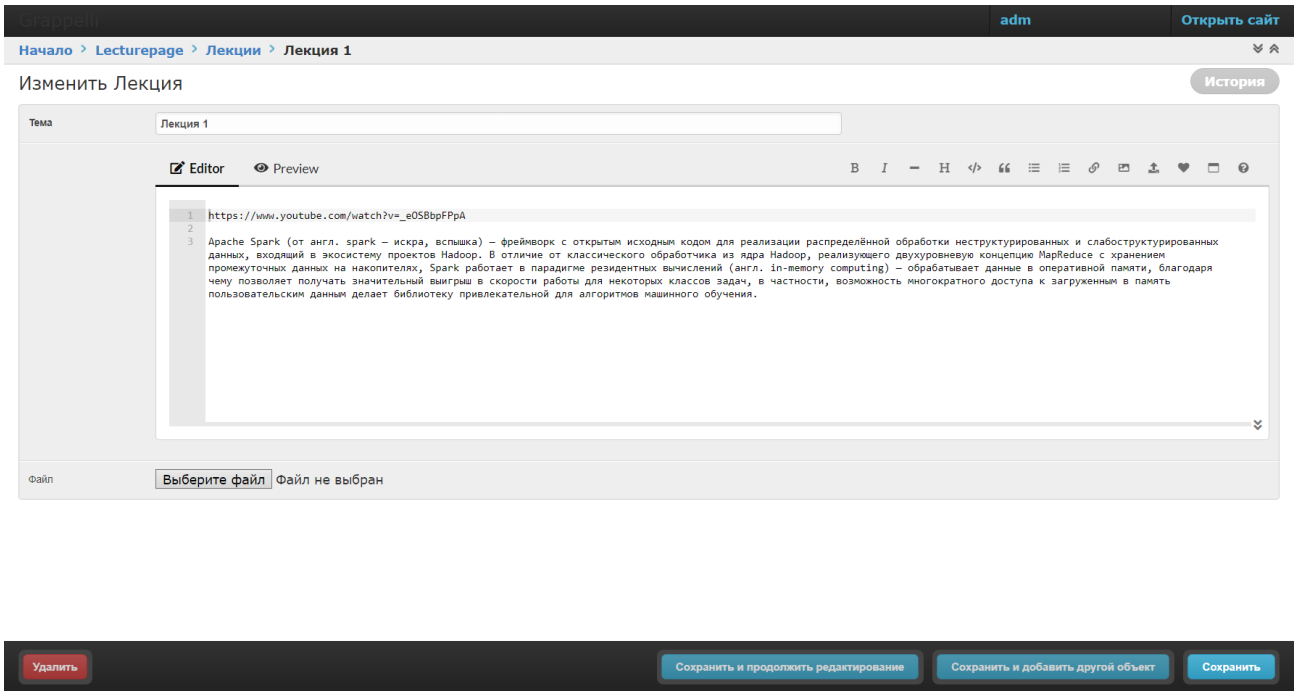


Рис. 2.13.

Стоит отметить что преподаватель, благодаря технологии markdown, может размещать лекционный материал в произвольном порядке, наполняя его мультимедийным контентом. Данная технология позволяет крайне прогрессивно подойти к редактированию многострочного текста, концепцию которого хорошо изложили Билл Скотт и Тереза Нейл в своей книге. ([6])

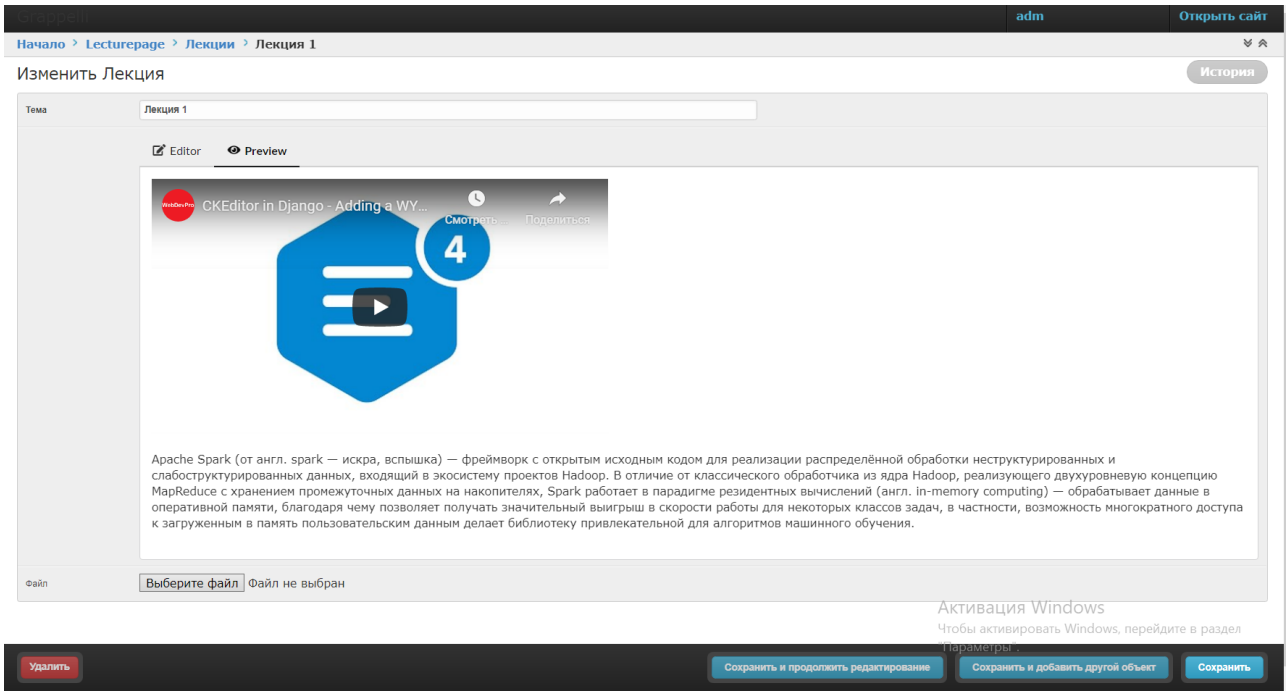


Рис. 2.14.

Одной из удобных функций редакторов лекционного материала и методического материала, является возможность предпросмотра страницы, с целью быстрой оценки того как лекционный материал будет отображаться в конечном итоге.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы был проведён анализ предметной области дисциплины «хранение обработка больших данных», и применимость к ней соответствующих доступных система онлайн обучения. При анализе предметной области были выявлены специфические черты дисциплины, а также определена необходимость в разработке системы мы для автоматической проверки кода. И встраивание этой системы в Веб-сервис обеспечивающие минимальный необходимый функционал для проведения обучения.

Также была разработана архитектура и прототип информационной системы с автоматической проверкой задач по анализу больших данных с помощью Apache Spark.

Система предоставляет минимальный необходимый пользовательский интерфейс и функции LMS систем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Хортон Уильям и Кэтрин Уильям и Кэтрин Хортон. - Электронное обучение: инструменты и технологии: Кудиц-образ, 2005 г.
2. Crockford Douglas the World's Most Misunderstood Programming Language Has Become the World's Most Popular Programming Language. - 2008.
3. Belida Balaji Varnasi и Sudha Introducing Maven. - [б.м.]: Apress, 2014 г.
4. George Nigel Build a website with Django 2 [Книга]. - [б.м.]: GNW Independent Publishing, August 27, 2018.
5. Билл Скотт Тереза Нейл "Проектирование веб-интерфейсов" [Книга]. - [б.м.] : Издательство O'REILLY,2011..
6. Халдеи Карау Энди Конвински, Патрик Венделл и Матей Захария Изучаем Spark МОЛНИЕНОСНЫЙ АНАЛИЗ ЛАННЫХ. - Москва: ДМК.