Министерство образования и науки Российской Федерации

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Факультет информатики
Кафедра программной инженерии

ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК

Руководитель ООП
д-р физ.-мат. наук, профессор
_____ О.А. Змеев
«10» июня 2016 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ КАМПУСНЫМИ КУРСАМИ

по основной образовательной программе подготовки магистров
«Управление проектами по разработке программного обеспечения»
направление подготовки
02.04.02 – Фундаментальная информатика и информационные технологии

Соколов Данила Александрович

Научный руководитель ВКР,
д-р физ.-мат. наук, профессор
_____ О.А.Змеев
подпись

«10» июня 2016 г.

Автор работы
студент группы №1447
_____ Д.А.Соколов
подпись

Томск 2016

РЕФЕРАТ

Выпускная квалификационная работа 48 стр., 26 рис., 12 источников.

DRUPAL, САЙТ, РАЗРАБОТКА МОДУЛЕЙ, РАЗРАБОТКА ТЕМ, ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ, CMS, АНАЛИЗ ТРЕБОВАНИЙ.

Цель работы – разработать и внедрить систему управления кампусными курсами ТГУ.

Результаты работы – разработан и внедрён информационный ресурс для управления кампусными курсами, в рамках данного ресурса было создано и запущено порядка 150 кампусных курсов.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Во времена тотальной информатизации современные университеты не могут позволить себе отставать от глобальных трендов в сфере развития применения информационных технологий внутри вуза. В качестве одного из таких примеров можно привести открытость вузов в сети Интернет. Это может выражаться следующими информационными ресурсами, которые могут быть как независимы друг от друга, так и являться частью одного и того же сайта:

- Главный сайт вуза и сайты учебных подразделений.
  На таких ресурсах обычно расположена информация, напрямую связанная с учебным процессом, а также с жизнью университета в целом, или подразделения в частности (новости, объявления, статичные страницы, документы, содержащие формальную информацию).
- Ресурсы, посвящённые неучебным подразделениям.
  В качестве таких подразделений можно привести управления информатизации, профсоюзы и прочие. Данные ресурсы, в первую очередь, создаются для внутреннего использования, но всё равно вносят свой вклад в общую информационную среду университета.
- Порталы и сервисы.
  Данные ресурсы, чаще всего, являются интерактивными (позволяют пользователю выполнять действия помимо поиска и просмотра контента) и могут быть направлены как на внутреннего, так и на внешнего пользователя.

Это основные примеры ресурсов, которые распространены в web-пространстве вузов. Среди этих ресурсов зачастую можно обнаружить и те, что посвящены дистанционному и дополнительному образованию, которые зачастую идут парами, но далеко не всегда. Примером дополнительного образования в рамках вуза целиком могут служить так называемые «кампусные курсы» (далее КК) — учебные курсы по различным направлениям подготовки, которые могут изучать студенты любых факультетов/курсов/образовательных программ.

Кампусные курсы не привязываются к факультету, а ведутся как бы от лица самого вуза, но при этом могут приносить пользу помимо простого получения знаний, например, можно получить сертификат о прохождении того или иного курса, который в последствии может быть предъявлен работодателю, или в учебном заведении (например, для закрытия схожего по содержанию предмета).

Так как ТГУ активно занимается модернизацией своего информационного пространства и модели образования, то кампусные курсы были запущены в рамках программы повышения конкурентоспособности 5-100. И, как следствие, возникла необходимость в разработке информационного ресурса, позволяющего студентам быть в курсе о существующих кампусных курсах, их новостях и иметь возможность записываться на данные курсы.

Целью данной работы является разработка и внедрение системы управления кампусными курсами ТГУ.

Задачи представлены ниже:

- Собрать и формализовать требования к разрабатываемой системе.
- Спроектировать разрабатываемую систему.
- Разработать информационную систему управления кампусными курсами.
- Протестировать и внедрить разработанную систему.

Представленные выше задачи будут подробно расписаны в последующих главах.

1 Определение и фиксация требований

Данная глава посвящена сбору и формализации требований. Так как разрабатываемая система имеет конкретного заказчика, то работы по сбору требований велись именно с ним.

Представленные ниже требования разбиты на функциональные и нефункциональные. Изначально требования представляются в том виде, в котором их указал заказчик. Далее же требования были обработаны, переформулированы и формализованы в виде диаграмм и сценариев вариантов использования.

Предлагается начать представление требований с нефункциональных, как менее объёмных.

1.1 Нефункциональные требования

Согласно полученной информации от заказчика, а также следуя политике управления информатизации ТГУ, разрабатываемый информационный ресурс должен:

- Поддерживать авторизацию преподавателей через систему accounts.tsu.ru.
- Собирать о пользователях разрабатываемой системы с доступных сервисов корпоративной среды ТГУ в автоматическом режиме следующую информацию:
  - ФИО.
  - Место работы.
  - Должность.
  - Учёная степень.
  - Учёное звание.
  - Email.
- Быть разработан как подсистема сайта «центра развития качества образования» ТГУ, реализованного на CMS Drupal

1.2 Функциональные требования

В данном разделе представлены функциональные требования к разрабатываемой системе, разбитые по ролям пользователей. Общий функционал

Разрабатываемый информационный ресурс должен обеспечивать:

- Возможность просмотра общедоступной информации.
- Возможность авторизации на сайте.

1.2.1 Функциональные возможности Студента

Для авторизованных пользователей, чья роль определяется как студент, web-приложение должно обеспечивать следующий функционал:

- Возможность записаться на КК.
- Возможность отменить запись на КК.
- Возможность задать вопрос администратору сайта.
- Возможность перейти из раздела «Мои курсы» на страницы КК.

На странице КК:

- Возможность написать отзыв о КК.
- Возможность распечатать предзаполненный макет сертификата о завершении КК.
- Возможность ознакомиться и скачать материалы, размещенные на странице КК преподавателем.
- Возможность перейти на страницу курса в СДО Moodle, по ссылке, предоставленной преподавателем.

1.2.2 Функциональные возможности Преподавателя

Для авторизованных пользователей, чья роль определяется как преподаватель, web-приложение должно обеспечивать следующий функционал:

- Возможность перейти из раздела «Мои курсы» на страницы КК, автором которых он зарегистрирован в системе КК.

На странице КК:

- Возможность загружать/обновлять в определенном для этого разделе (Мои курсы) «Рабочую программу КК» (с доступом только для администратора сайта).
- Возможность загружать/обновлять в определенном для этого разделе «Ссылку на курс в СДО Moodle».
- Возможность размещать на странице КК разнообразный контент, обеспечивающий организационно-методическую поддержку КК (текстовые/аудио/видео-файлы (для скачивания или стриминг с сторонних сервисов (Youtube), ссылки, иллюстрации, и пр.).
- Возможность загружать на свой рабочий компьютер в формате excel ряд рабочих документов (списки групп, табель посещаемости, зачётную ведомость) с данными доступными из web-приложения.
- Возможность создавать объявления по курсу и делать автоматическую рассылку на адреса электронной почты всех студентов, зачисленных на данный КК, с автоматическим копированием сообщения на почту администратора с указанием сведений об объявлении (от кого; кому; тема (при наличии); текст объявления; дата отправки).
- Возможность отметить и редактировать в предзаполненной электронной форме отчета по курсу – результаты прохождения студентами контрольной точки («Аттестован»/«Не аттестован») и промежуточной аттестации по КК («Зачет», «Не зачет», «Не явился на зачет») – с возможностью редактирования до определенного администратора срока, а также заполнить графу «Элективный/факультативный курс» по предъявлению индивидуального учебного плана (ИУП) студента.

1.2.3 Функциональные возможности Администратора

Для авторизованных пользователей, чья роль определяется как администратор, web-приложение должно обеспечивать следующий функционал:

- Возможность формировать различные формы статистики. Вопрос о внесении правок в готовые формы или добавление новых решается отдельно в зависимости от сложности.
- Возможность заполнения каталога КК

– добавление, редактирование, удаление авторов КК;

– добавление, редактирование, удаление курсов и информации, содержащейся в них.

- Возможность выбирать из списка КК те, которые необходимо
  - открыть для записи;
  - отметить примечанием «Запись не ведется» как неактивные в данном семестре;
  - переместить в архив каталога (скрытый для пользователей);
  - достать КК из архива и открыть для записи/поставить отметку «Запись не ведется»;
- Возможность размещать на страницах КК дополнительную информацию (инструкции, анкеты, опросы, уведомления).
- Возможность ограничения времени, отведенного для редактирования преподавателем электронной формы с результатами прохождения студентами контрольной точки («Аттестован»/«Не аттестован») и промежуточной аттестации по КК («Зачет», «Не зачет», «Не явился на зачет») - ограничения отдельно: по времени заполнения контрольной точки и внесения результатов зачета, возможность снятия данного ограничения.

При осуществлении записи на КК и формировании учебных групп:

- Возможность вручную добавить\удалить определенного студента в\из учебную\ой группы по согласованию с преподавателем.
- Сформировать списки учебных групп на текущий семестр с
  - данными о курсе (Название, кол-во зачетных единиц) и преподавателе/лях (ФИО, место работы, должность);
  - данными о студентах, доступными из веб-приложения.

1.3 Формализация и анализ требований

После того, как требования, предоставленные заказчиком, были собраны и обработаны, была проведена работа по их формализации – приведении к одному из стандартов их представления – диаграммам и сценариям вариантов использования.

В требованиях к разрабатываемой системе были упомянуты три роли пользователей: «студент», «преподаватель», «администратор». Также, существует и четвёртая роль – «гость», но весь функционал пользователей с этой ролью ограничивается просмотром, поэтому на диаграммах вариантов

использования он не описывается, также отсутствуют подобные варианты использования.

При этом, основной функционал пользователей «преподаватель» и «студент» никак не пересекается, а пользователи с ролью «администратор» имеют доступ ко всем функциям, доступным как пользователям «студент», так и «преподаватель». Это сделано из следующих соображений:

- Администратор может помогать пользователям, в случае возникновения у них проблем.

Далее рассмотрим варианты использования пользователя с ролью «администратор».

Рисунок 1 – Диаграмма вариантов использования актёра «Администратор»

На диаграмме представлены все значимые варианты использования актёра «Администратор». Далее будут рассмотрены сценарии архитектурно значимых вариантов использования для актёра «Администратор» в форме текстовой спецификации Алистера Кобёрна [5].

После каждого описания варианта использования следует диаграмма коммуникаций анализа соответствующего варианта использования.

Таблица 1 – Описание и сценарий варианта использования «Создать курс»

Название ВИ	**Создать курс**
Цель	Создать курс
Актёр	Администратор сайта
Предусловия	Нет
Сценарий	1. Пользователь переходит на административную страницу «Добавить содержимое». 2. Система выводит список доступных для создания типов материалов. 3. Пользователь выбирает типа материала «Курс». 4. Система открывает окно «Создание материала Курс». 5. Пользователь вводит обязательную для ввода информацию. 6. Система выполняет верификацию данных. 7. Система сохраняет курс в базу данных.
Расширения	5.1. Введённые пользователем данные не прошли валидацию. 5.1.1. Система открывает окно «Создание материала Курс» с введёнными ранее пользователем данными и отмечает, какие данные введены некорректно или пропущены. 5.1.2. Пользователь исправляет введённые данные.

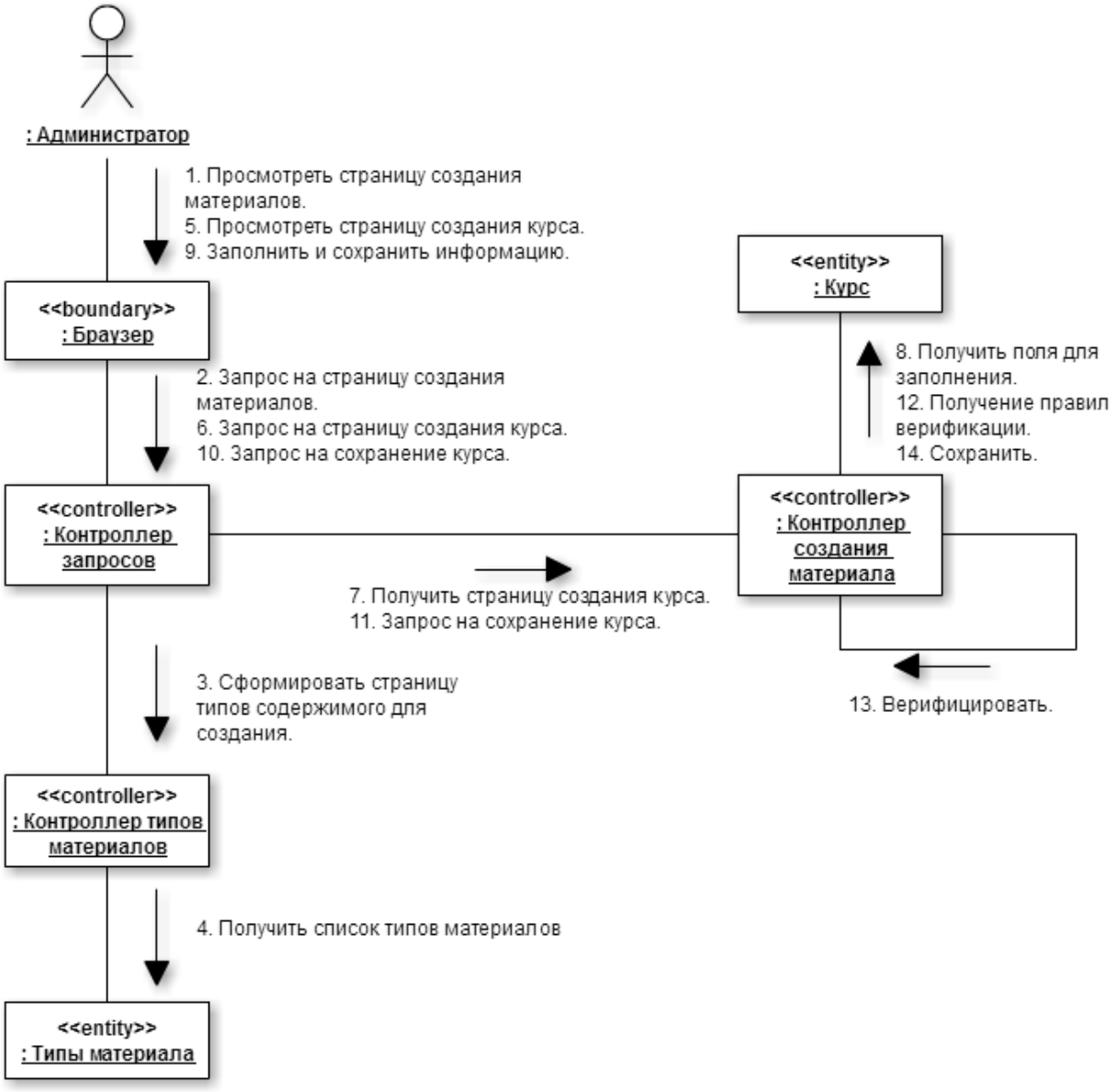


Рисунок 2 – Диаграмма коммуникаций анализа варианта использования «создать курс»

Таблица 2 – Описание и сценарий варианта использования «Управлять состоянием курса»

Название ВИ	**Управлять состоянием курса**
Цель	Управлять состоянием курса, изменять его параметры
Актёр	Администратор сайта
Предусловия	1. В системе должны быть созданы курсы. 2. Пользователь находится в разделе «кампусные курсы» на сайте.
Сценарий	1. Пользователь переходит на страницу «Управление курсами».

	2. Система выводит список созданных курсов. 3. Пользователь модифицирует настройки курса и нажимает «сохранить». 4. Система сохраняет изменения курса в базу данных.
Альтернативный сценарий 1	1. Пользователь переходит на страницу каталога курсов. 2. Система выводит список существующих курсов. 3. Пользователь выбирает курс. 4. Система перенаправляет пользователя на страницу выбранного курса. 5. Пользователь выбирает режим «правка». 6. Система выдаёт страницу или всплывающее окно с формами правки курса. 7. Пользователь изменяет настройки состояния курса и сохраняет изменения. 8. Система записывает изменения курса в базу данных.
Расширения альтернативного сценария 1	3.1. Пользователь использует систему фильтрации курсов для облегчения поиска. 3.1.1. Система выдаёт отфильтрованный список курсов. 3.1.2. Пользователь выбирает курс.
Альтернативный сценарий 2	1. Пользователь переходит на административную страницу «содержимое». 2. Система выводит все материалы, созданные на сайте. 3. Пользователь выбирает курс. 4. Система перенаправляет пользователя на страницу выбранного курса. 5. Пользователь выбирает режим «правка». 6. Система выдаёт страницу или всплывающее окно с формами правки курса. 7. Пользователь изменяет настройки состояния курса и сохраняет изменения. 8. Система записывает изменения курса в базу данных.
Расширения альтернативного сценария 2	3.1. Пользователь использует систему фильтрации материалов для облегчения поиска. 3.1.1. Система выдаёт отфильтрованный список материалов. 3.1.2. Пользователь выбирает курс.

Рисунок 3 – Диаграмма коммуникаций анализа варианта использования «управлять состоянием курса»

Таблица 3 – Описание и сценарий варианта использования «Управлять списком студентов курса»

Название ВИ	**Управлять списком студентов курса**
Цель	Изменять список студентов, обучающихся на курсе
Актёр	Администратор сайта
Предусловия	1. В системе должны быть созданы курсы. 2. Пользователь находится в разделе «кампусные курсы» на сайте.
Сценарий	1. Пользователь переходит на страницу каталога курсов. 2. Система выводит список существующих курсов. 3. Пользователь выбирает курс и инициирует переход на внутреннюю страницу курса. 4. Система перенаправляет пользователя на внутреннюю страницу выбранного курса. 5. Пользователь выбирает режим «добавить/удалить студента». 6. Система выдаёт страницу или всплывающее окно с формами правки группы студентов курса. 7. Пользователь вносит изменения в список группы студентов курса и сохраняет изменения. 8. Система записывает изменения курса в базу данных.
Расширения	3.1. Пользователь использует систему фильтрации курсов для облегчения поиска. 3.1.1. Система выдаёт отфильтрованный список курсов. 3.1.2. Пользователь выбирает курс и инициирует переход на внутреннюю страницу курса.
Альтернативный сценарий 1	1. Пользователь переходит на страницу «Управление курсами». 2. Система выводит список созданных курсов. 3. Пользователь выбирает курс из списка. 4. Система перенаправляет пользователя на внутреннюю страницу выбранного курса. 5. Пользователь выбирает режим «добавить/удалить студента». 6. Система выдаёт страницу или всплывающее окно с формами правки группы студентов курса. 7. Пользователь вносит изменения в список группы студентов курса и сохраняет изменения. 8. Система записывает изменения курса в базу данных.

Альтернативный сценарий 2	1. Пользователь переходит на административную страницу «содержимое». 2. Система выводит все материалы, созданные на сайте. 3. Пользователь выбирает курс. 4. Система перенаправляет пользователя на страницу выбранного курса. 5. Пользователь инициирует переход на внутреннюю страницу курса. 6. Система перенаправляет пользователя на внутреннюю страницу выбранного курса. 7. Пользователь выбирает режим «добавить/удалить студента». 8. Система выдаёт страницу или всплывающее окно с формами правки группы студентов курса. 9. Пользователь вносит изменения в список группы студентов курса и сохраняет изменения. 10. Система записывает изменения курса в базу данных.
Расширения альтернативного сценария 2	3.1. Пользователь использует систему фильтрации материалов для облегчения поиска. 3.1.1. Система выдаёт отфильтрованный список материалов. 3.1.2. Пользователь выбирает курс.

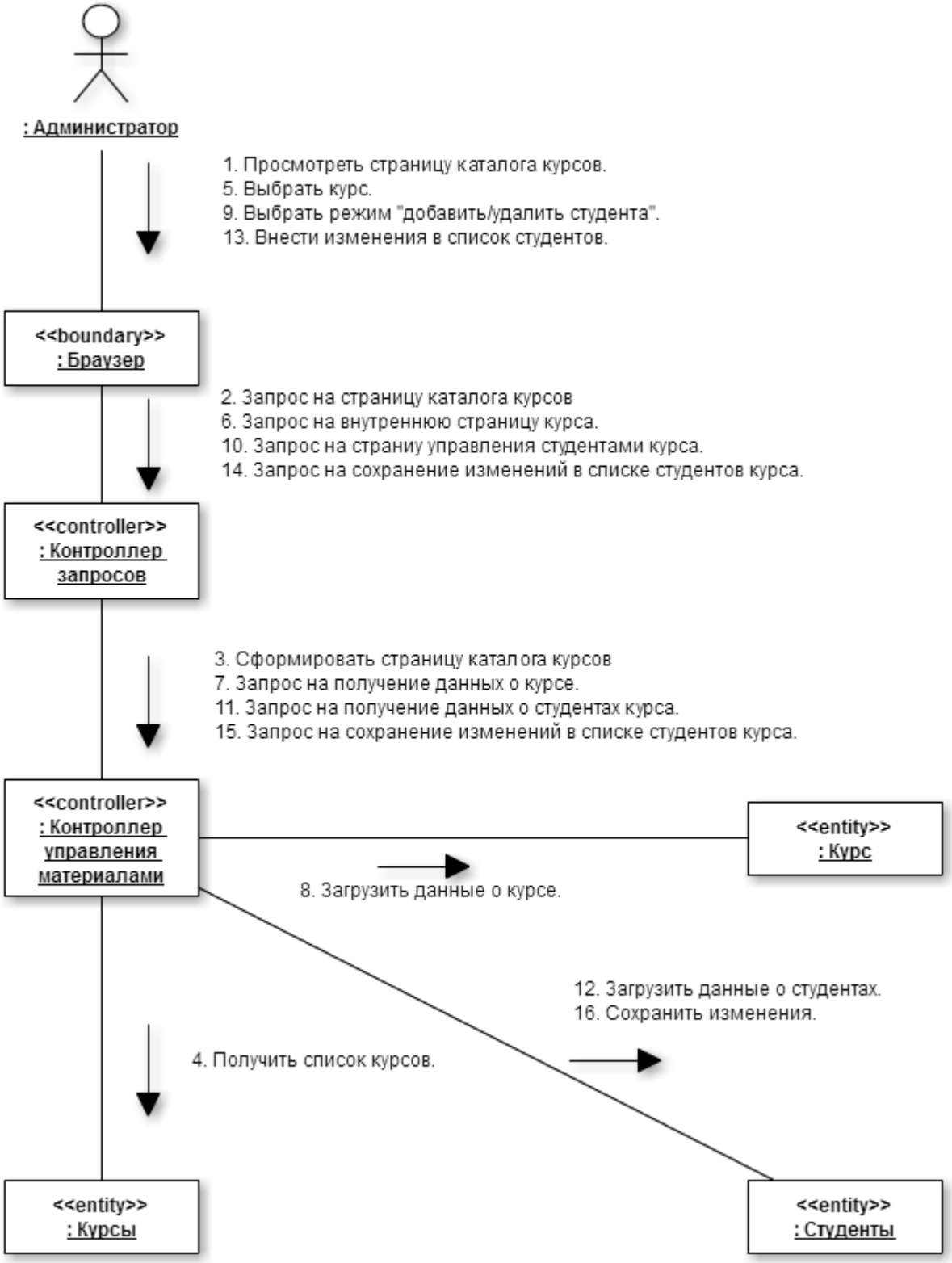


Рисунок 4 – Диаграмма коммуникаций анализа варианта использования «Управлять списком студентов курса»

Далее рассмотрим варианты использования для пользователя с ролью «Преподаватель».

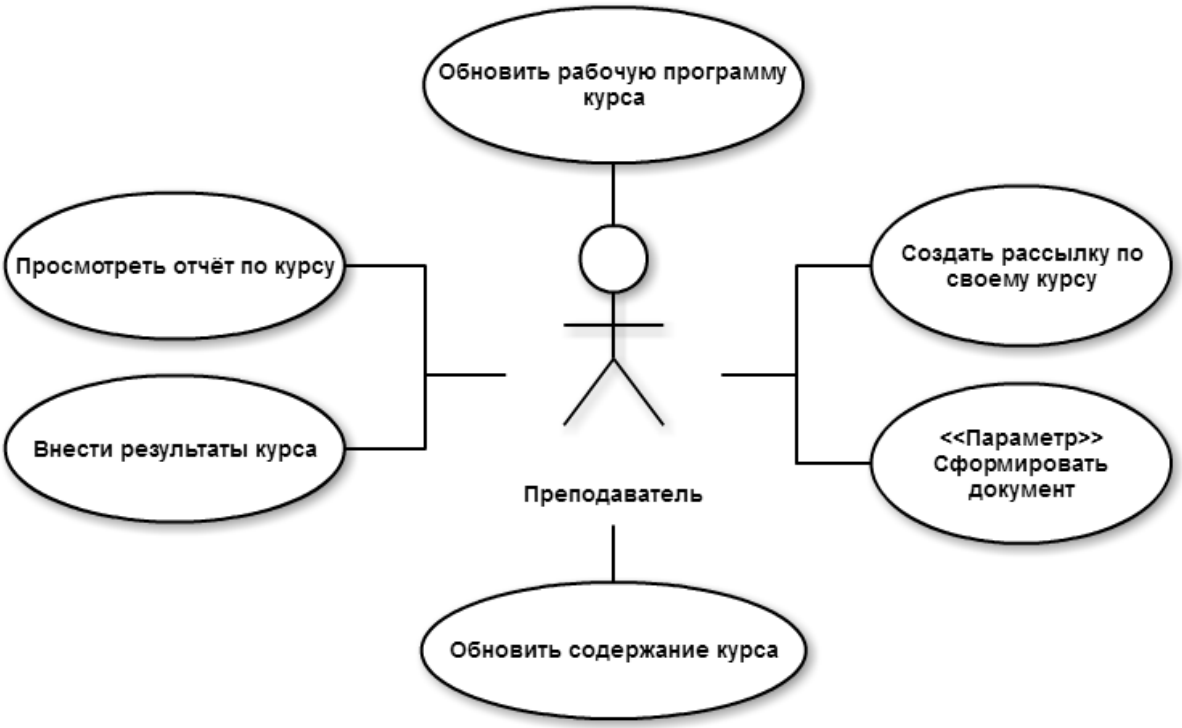


Рисунок 5 – Диаграмма вариантов использования актёра «Преподаватель»

Таблица 4 – Описание и сценарий варианта использования «Обновить содержание курса»

Название ВИ	**Обновить содержание курса**
Цель	Изменять параметры курса
Актёр	Преподаватель
Предусловия	1. Пользователь должен числиться преподавателем хотя бы одного курса.
Сценарий	1. Пользователь инициирует переход на страницу личного кабинета. 2. Система открывает страницу личного кабинета пользователя. 3. Пользователь инициирует переход на страницу «мои курсы». 4. Система выводит список существующих курсов пользователя. 5. Пользователь выбирает курс и инициирует переход на внутреннюю страницу курса. 6. Система перенаправляет пользователя на внутреннюю страницу выбранного курса. 7. Пользователь выбирает режим «редактировать курс». 8. Система выдаёт страницу или всплывающее окно с формами правки информации о курсе.

	9. Пользователь вносит изменения в описание курса и сохраняет изменения. 10. Система проводит процедуру верификации данных. 11. Система записывает изменения курса в базу данных.
Расширения	10.1. Введённые пользователем данные не прошли валидацию. 10.1.1. Система открывает страницу редактирования курса с введёнными ранее данными. 10.1.2. Пользователь вносит изменения во внесённые ранее данные. 10.1.3. Система проводит процедуру верификации данных. 10.1.4. Система записывает изменения курса в базу данных.
Альтернативный сценарий 1	1. Пользователь переходит на страницу каталога курсов. 2. Система выводит список существующих курсов. 3. Пользователь выбирает курс и инициирует переход на внутреннюю страницу курса. 4. Система перенаправляет пользователя на внутреннюю страницу выбранного курса. 5. Пользователь выбирает режим «редактировать курс». 6. Система выдаёт страницу или всплывающее окно с формами правки информации о курсе. 7. Пользователь вносит изменения в описание курса и сохраняет изменения. 8. Система записывает изменения курса в базу данных.
Расширения альтернативного сценария 1	3.1. Пользователь использует систему фильтрации курсов для облегчения поиска. 3.1.1. Система выдаёт отфильтрованный список курсов. 3.1.2. Пользователь выбирает курс и инициирует переход на внутреннюю страницу курса.

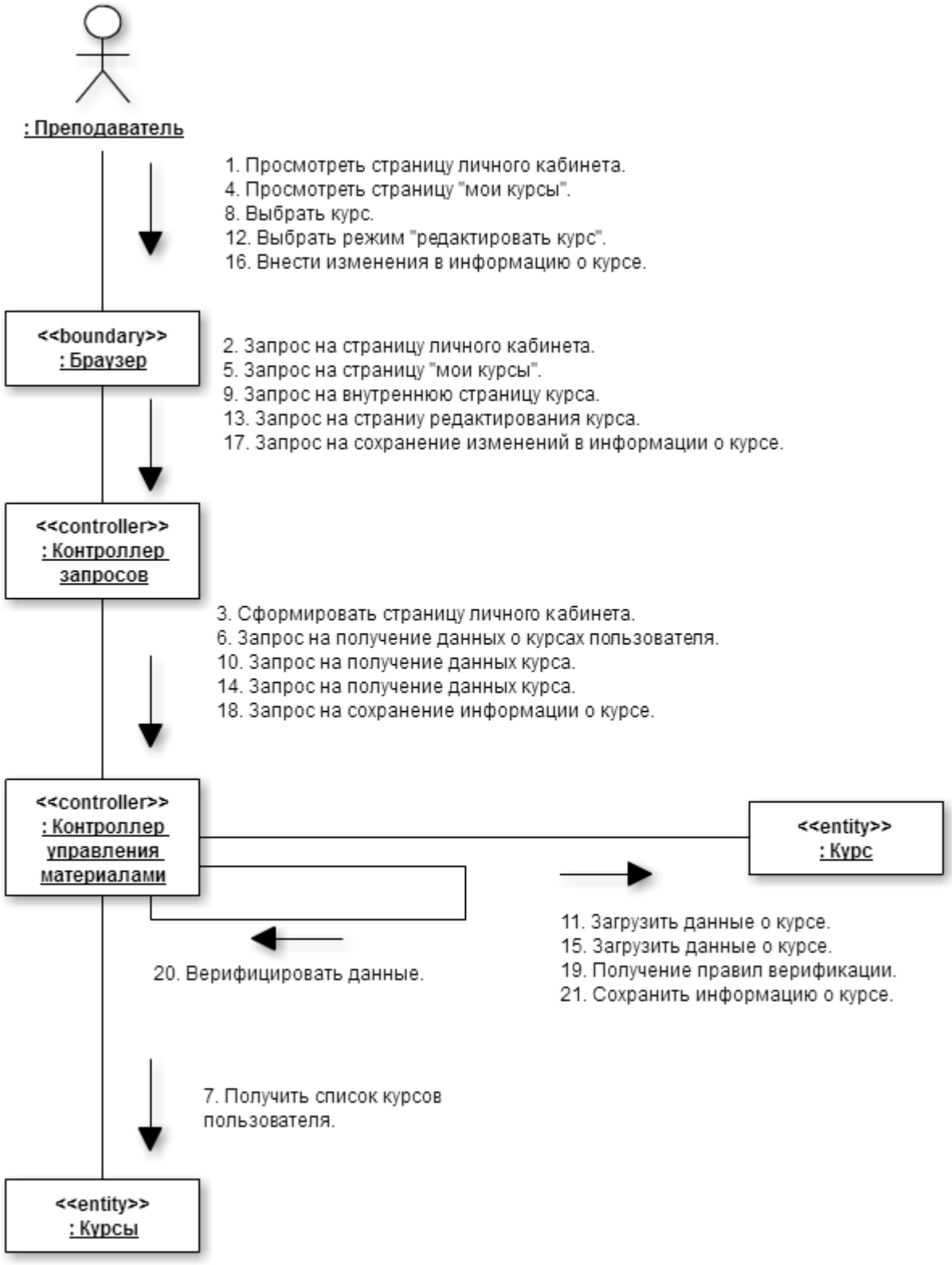


Рисунок 6 – Диаграмма коммуникаций анализа варианта использования «Обновить содержание курса»

В завершение рассмотрим варианты использования для пользователей с ролью «Студент».

Рисунок 7 – Диаграмма вариантов использования актёра «Студент»

Таблица 5 – Описание и сценарий варианта использования «Записаться на курс»

Название ВИ	**Записаться на курс**
Цель	Записаться на курс
Актёр	Студент
Предусловия	Нет
Сценарий	1. Пользователь переходит на страницу каталога курсов. 2. Система выводит список существующих курсов. 3. Пользователь выбирает курс и инициирует запись на курс. 4. Система проверяет, есть ли свободные места. 5. Система прикрепляет студента к курсу. 6. Система записывает изменения курса в базу данных.
Расширения	3.1. Пользователь использует систему фильтрации курсов для облегчения поиска. 3.1.1. Система выдаёт отфильтрованный список курсов. 3.1.2. Пользователь выбирает курс и инициирует запись на курс.

	4.1. Свободных мест в группе записи на курс не оказалось. 4.1.1. Система прикрепляет студента к листу ожидания. 4.1.2. Система записывает изменения курса в базу данных
Альтернативный сценарий 1	1. Пользователь переходит на страницу каталога курсов. 2. Система выводит список существующих курсов. 3. Пользователь выбирает курс и инициирует переход на страницу курса. 4. Система перенаправляет пользователя на страницу выбранного курса. 5. Пользователь инициирует запись на курс. 6. Система проверяет, есть ли свободные места. 7. Система прикрепляет студента к курсу. 8. Система записывает изменения курса в базу данных.
Расширения альтернативного сценария 1	3.1. Пользователь использует систему фильтрации курсов для облегчения поиска. 3.1.1. Система выдаёт отфильтрованный список курсов. 3.1.2. Пользователь выбирает курс и инициирует переход на внутреннюю страницу курса. 6.1. Свободных мест в группе записи на курс не оказалось. 6.1.1. Система прикрепляет студента к листу ожидания. 6.1.2. Система записывает изменения курса в базу данных

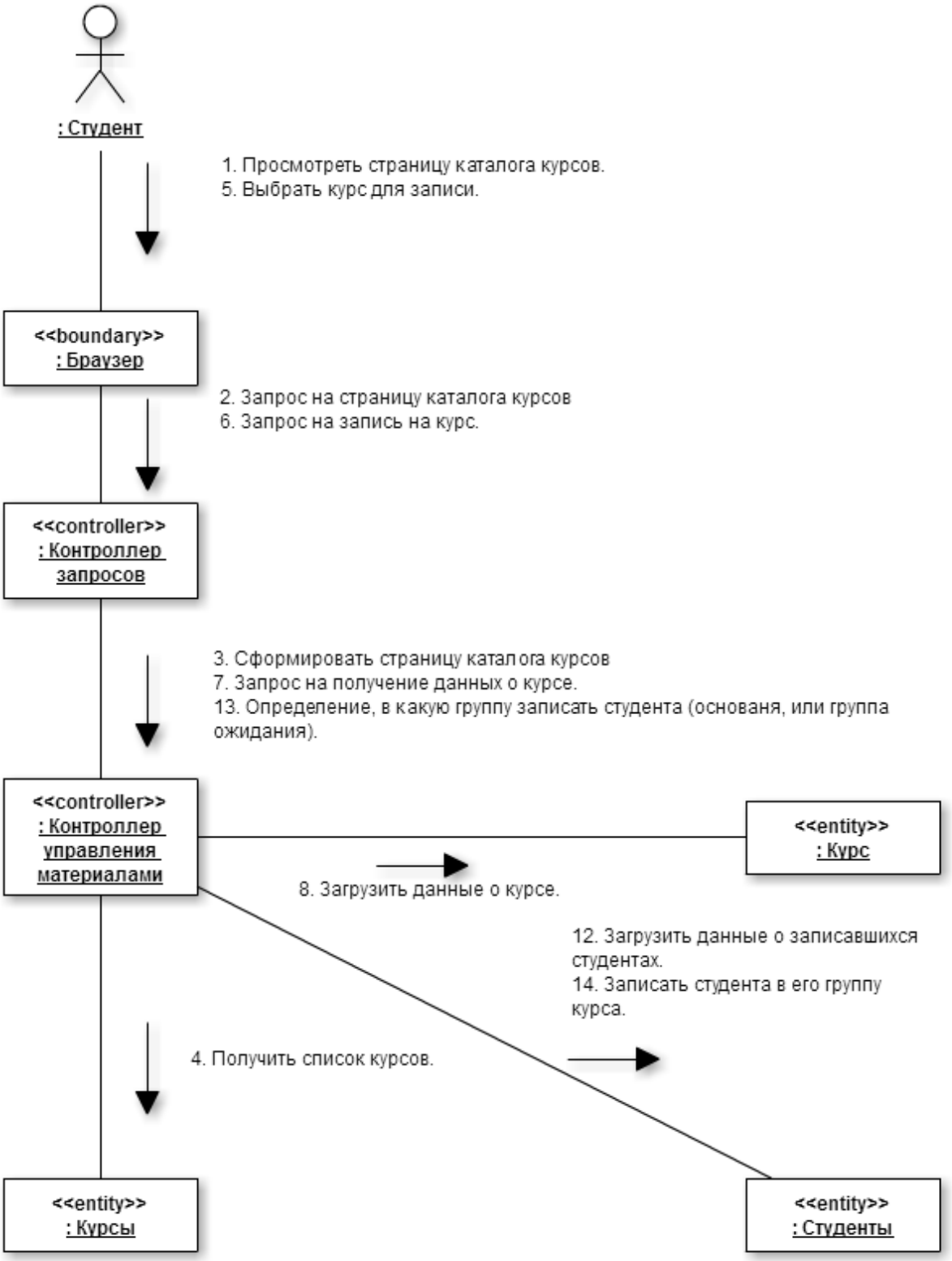


Рисунок 8 – Диаграмма коммуникаций анализа варианта использования «Записаться на курс»

Таблица 6 – Описание и сценарий варианта использования «Отписаться от курса»

Название ВИ	**Отписаться от курса**
Цель	Отписаться от курса
Актёр	Студент
Предусловия	Нет
Сценарий	1. Пользователь переходит на страницу личного кабинета. 2. Система выводит страницу личного кабинета пользователя. 3. Пользователь переходит на страницу «мои курсы». 4. Система выводит список существующих курсов пользователя. 5. Пользователь выбирает курс и инициирует отмену записи на курс. 6. Система записывает изменения курса в базу данных.
Расширения	6.1. Пользователь состоял в «листе ожидания» курса. 6.1.1. Система открепляет пользователя от листа ожидания. 6.1.2. Система сохраняет изменения курса в базу данных. 6.1.2.1. Пользователь был не единственным в листе ожидания. 6.1.2.1.1. Система сохраняет изменения в БД. 6.1.2.1.2. Система отправляет следующему пользователю из списка ожидание уведомление на почту о том, что он был зачислен в основную группу курса.
Альтернативный сценарий 1	1. Пользователь переходит на страницу каталога курсов. 2. Система выводит список существующих курсов. 3. Пользователь выбирает курс и инициирует переход на страницу курса. 4. Система перенаправляет пользователя на страницу выбранного курса. 7. Пользователь инициирует отмену записи на курс. 5. Система записывает изменения курса в базу данных.
Расширения альтернативного сценария 1	3.1. Пользователь использует систему фильтрации курсов для облегчения поиска. 3.1.1. Система выдаёт отфильтрованный список курсов.

	3.1.2. Пользователь выбирает курс и инициирует переход на внутреннюю страницу курса. 5.1. Пользователь состоял в «листе ожидания» курса. 5.1.1. Система открепляет пользователя от листа ожидания. 5.1.2. Система сохраняет изменения курса в базу данных. 5.1.2.1. Пользователь был не единственным в листе ожидания. 5.1.2.1.1. Система сохраняет изменения в БД. 5.1.2.1.2. Система отправляет следующему пользователю из списка ожидания уведомление на почту о том, что он был зачислен в основную группу курса.

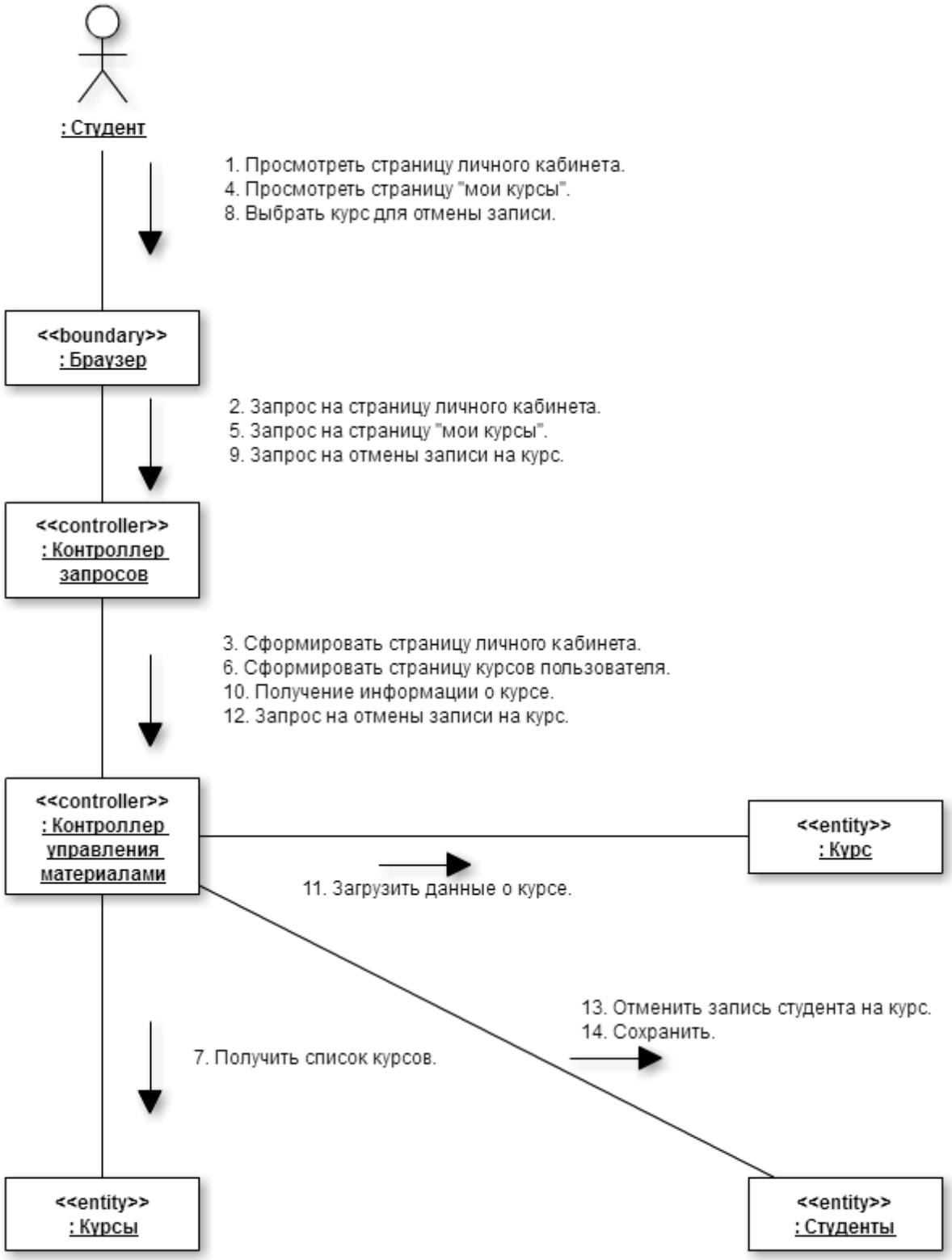


Рисунок 9 – Диаграмма коммуникаций анализа варианта использования «Отписаться от курса»

На основании построенных диаграмм анализа была спроектирована и зафиксирована диаграмма предметной области в форме диаграммы классов.

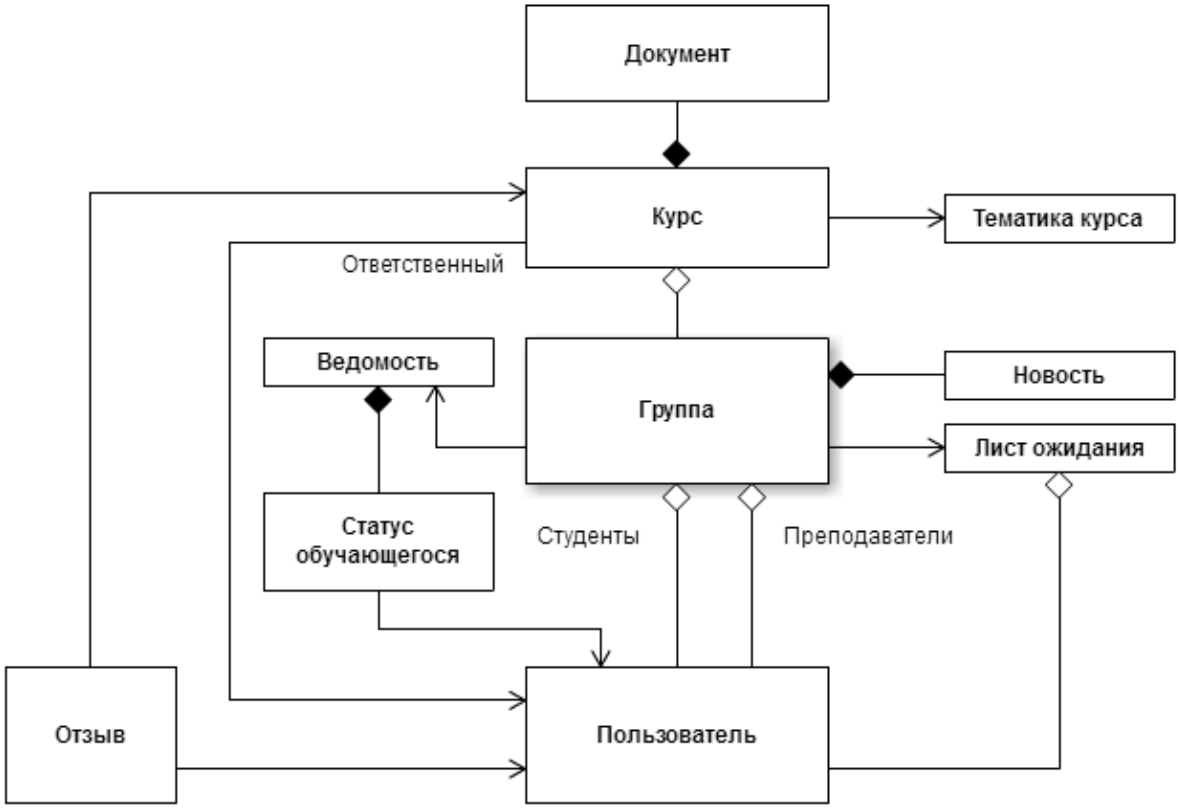


Рисунок 10 – Диаграмма классов модели предметной области

Данная диаграмма фиксирует предметную область разрабатываемой системы и не привязана к классам реализации. В последствии данная диаграмма может быть изменена под влиянием среды разработки (фреймворка).

2 Проектирование

2.1 Особенности архитектуры CMS Drupal

В качестве среды разработки была взята CMS Drupal [1, 2, 9], согласно приведённым в разделе 1.1. требованиям. Данная система является не только CMS (Content Management System, Система управления контентом), но и CMF (Content Management Framework). Комбинируя эти два качества Drupal представляет из себя очень гибкую и в то же время довольно простую и дружелюбную к пользователю среду для разработки сайтов.

В основе Drupal лежит архитектурный паттерн [7] PAC – Presentation-Abstraction-Control (Представление-Абстракция-Управление) [3, 4]. Данный паттерн по идеологии близок к MVC за счёт того, что разделяет систему на три типа компонент, ответственных за специфичные аспекты функционала приложения.

- Компонент «Abstraction» отвечает за работу с данными и схож с компонентом «Model» в MVC.
- Компонент «Presentation» отвечает за форматирование аудиовизуального представления данных и схож с компонентом «View» в MVC.
- Компонент «Control» отвечает за обработку потока управления и коммуникаций между двумя предыдущими компонентами и схож с компонентом «Controller» в MVC.

В отличие от MVC, PAC использует иерархическую структуру агентов, каждый из которых состоит из троек «представление-абстракция-управление». Агенты взаимодействуют только с помощью компонента «управление». В отличие от некоторых реализаций MVC, не только «абстракция» не знает о «представлении», но и наоборот.

В качестве классического примера данной архитектуры можно привести следующий пример: система контроля воздушного траффика. Каждый агент получает входные данные с радара о локации самолёта и использует компонент «представление» для отрисовки изображения самолёта на экране. Другой же агент независимо получает входные данные о другом самолёте и также рисует его на экране. Другие агенты выводят информацию о погоде, рейсах и т.д.

PAC также называют «иерархичная MVC».

Пример паттерна PAC представлен на концептуальной диаграмме ниже.

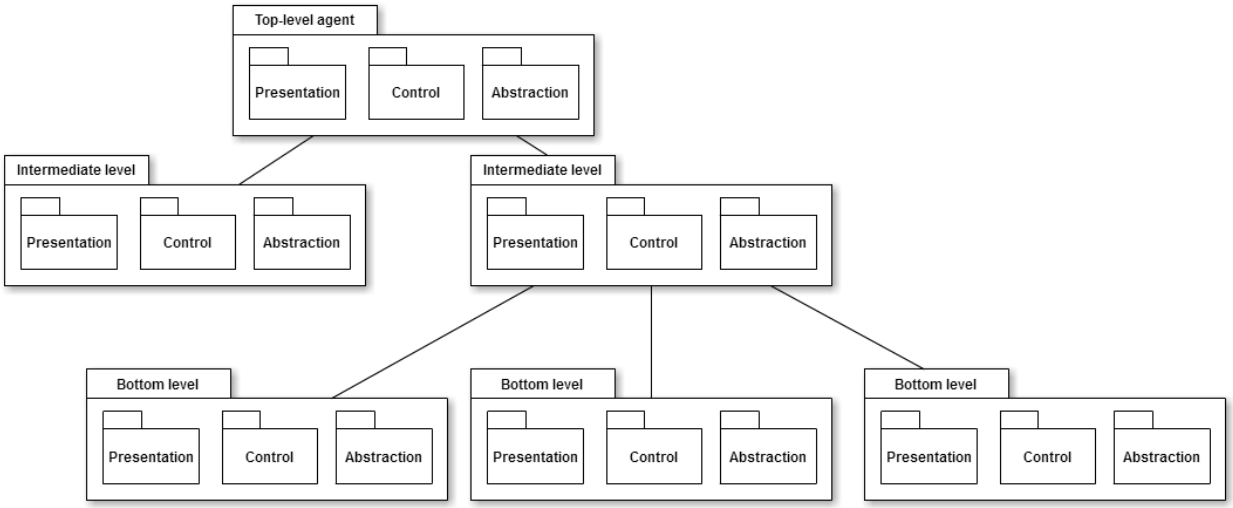


Рисунок 11 – Концептуальная диаграмма, демонстрирующая устройство паттерна PAC

Если попытаться представить архитектуру Drupal в рамках идеологии PAC, то концептуально получается нечто подобное диаграмме ниже.

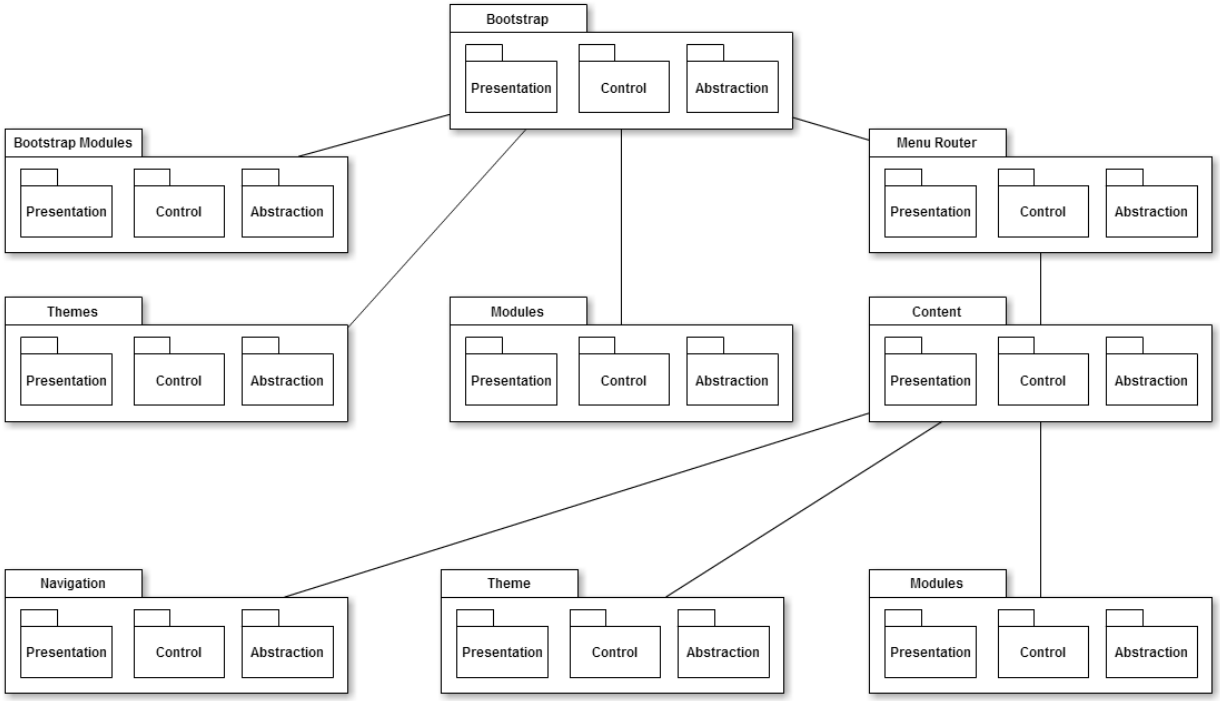


Рисунок 12 – Концептуальная диаграмма, демонстрирующая устройство Drupal в идеологии паттерна PAC

Особо значимыми тут являются два пакета – Bootstrap и Menu Router.

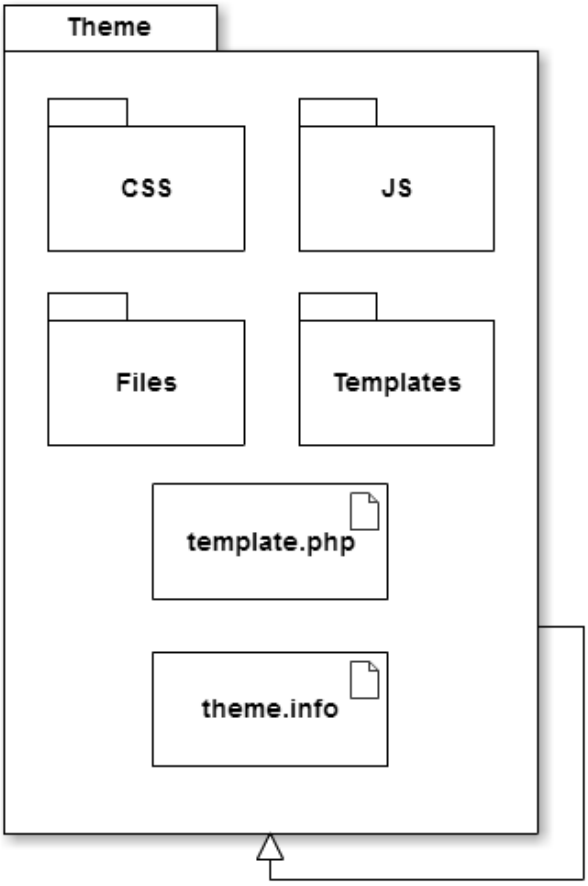
Любой запрос к системе проходит через пакет Bootstrap и, в зависимости от и настроек, компоненты Bootstrap вызывают следующие в иерархии объекты с необходимыми параметрами. Он же отвечает за загрузку модулей ядра, так же модулей, прописанных в автозагрузке. Многие из объектов, которые подключает Bootstrap просто инициализируются, а использоваться уже будут в иерархии ниже.

В пакете Menu Router происходит детальный разбор запроса и его маршрутизация, определяющая, в первую очередь, какой контент должен быть отдан в ответ на запрос, а также какие объекты должны быть также вызваны вместе с контентом.

2.2 Темизация в CMS Drupal

Далее рассмотрим систему темизации Drupal, а конкретно, физическое устройство темы. Тема здесь – набор правил и шаблонов, отвечающих за форматирование выводимой на экран информации.

На рисунке 13 изображено физическое устройство темы, отображающее наиболее важные компоненты. Стоит заметить, что одна тема может расширять другую, что значит, что всегда можно взять за основу одну тему и, не меняя её исходный код (что даёт возможность получать обновления родительской темы в автоматическом режиме, без необходимости ручной синхронизации), внести изменения, требующиеся при разработке сайта.



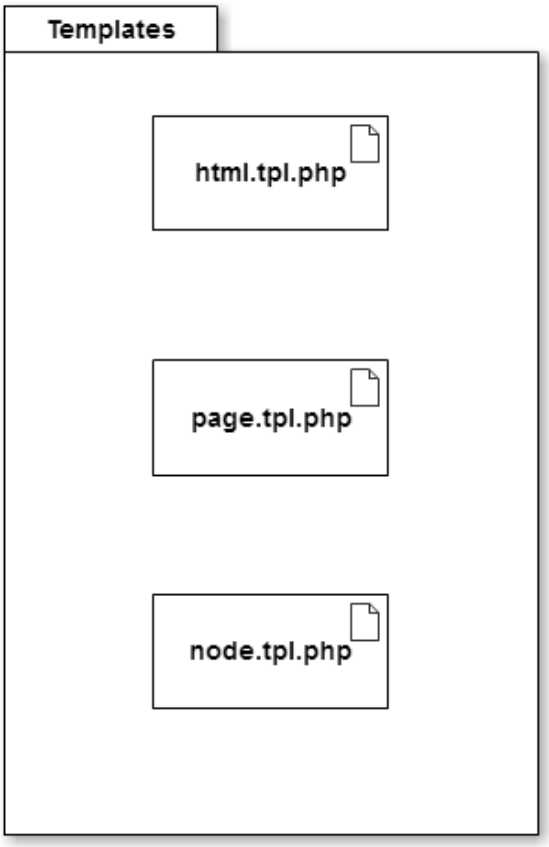


Рисунок 13 – физическое устройство системы темизации

Рисунок 14 – физическое устройство папки Template внутри папки темы

В файле **template.php** описываются функции, которые будут доступны в рамках всей темы (вспомогательные функции) и реализуются **хуки**, которые могут позволить изменить определённое по-умолчанию в системе поведение для тех или иных случаев.

Хук (или Hook) – это некая php-функция, имеющая шаблонную сигнатуру, использующаяся как точка расширения в функциональном мире Drupal. В качестве примера можно привести **hook_menu** – хук, использующийся для создания страниц (регистрация путей). При его реализации, вместо «hook» пишется имя того модуля, который его реализует, что позволит системе обнаружить и вызывать данную функцию при необходимости.

Файл **theme.info** имеет шаблонное наименование (вместо «theme» пишется машинное имя темы, такое же, как и имя корневой папки темы) и описывает параметры темы, такие как её машинное и человекочитаемое имя, требования к системе и зависимости. Кроме того, в данном файле регистрируются файлы стилей и скриптов, определяются будущие регионы

(области, в которых может быть выведен произвольный блок с контентом) темы.

Папки CSS, JS, FILES являются вспомогательными и могут быть устроены любым образом, ведь итоговое подключение этих файлов идёт в **theme.info** или в любом файле шаблона напрямую.

Далее более подробно рассмотрим содержимое папки **Templates**. В ней как раз и хранятся те самые файлы шаблонов, которые и форматируют выводимый на экран контент.

Первый из представленных файлов – **html.tpl.php**. Данный шаблон отвечает за HTML-код верхнего уровня, включающий в себя метаданные (в том числе подключение скриптов и стилей).

Второй и наиболее значимый файл – **page.tpl.php**. Это, так называемая, «Мастер-страница», или layout. Тут указывается основной шаблон всех страниц по-умолчанию. Тут же определяется вызов регионов, в которые система будет подставлять контент (например, регион «контент» для содержимой страницы, или «футер» для информации внизу страницы).

Остальные файлы, обычно, контент-ориентированы – они переопределяют вывод определённого вида контента, добавляя ему обёртки, или переупорядочивая содержимое. В качестве примера приведён файл **node.tpl.php**, использующийся для форматирования нод (конкретных страниц, материалов).

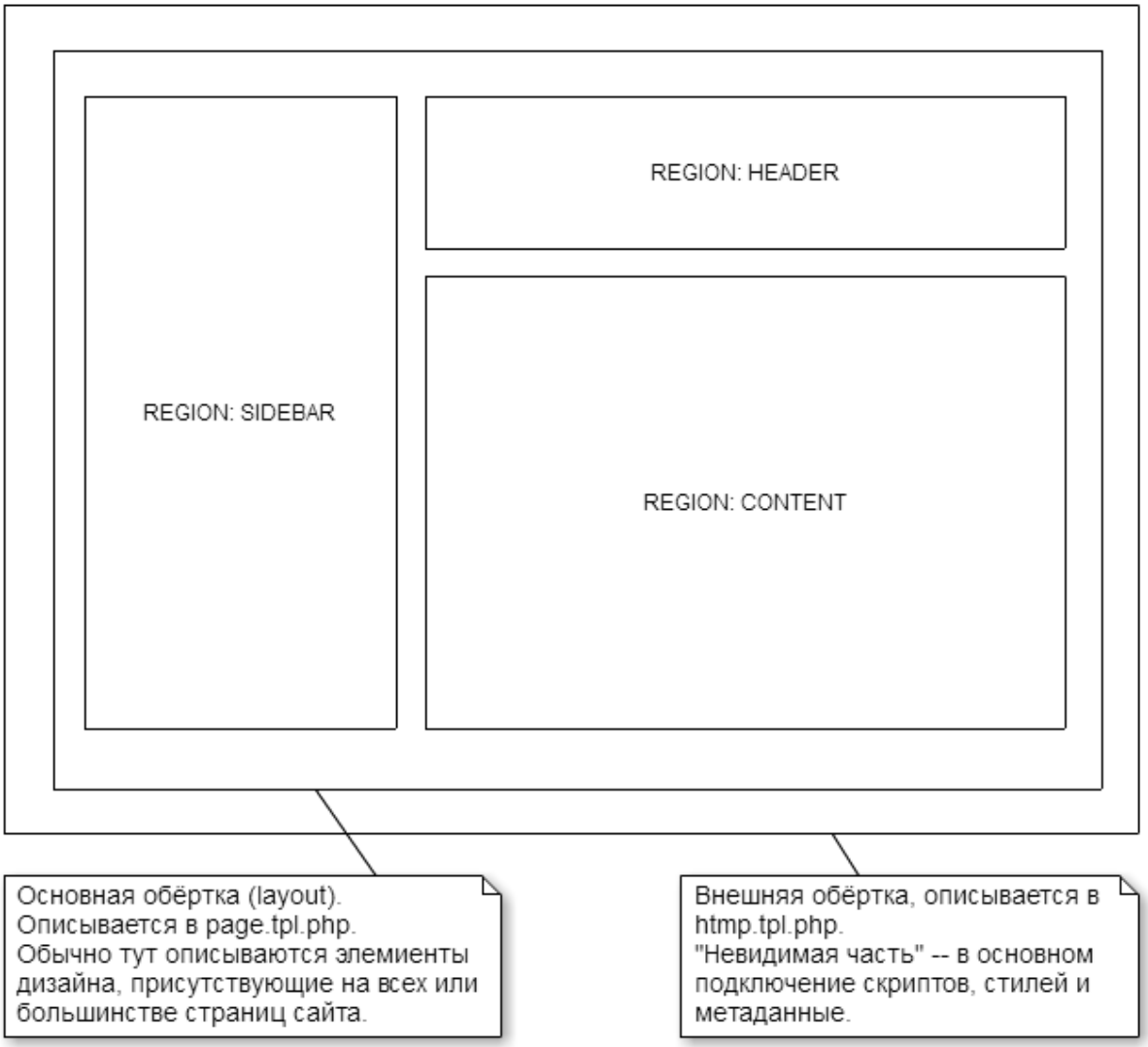


Рисунок 15 – Регионы и обёртки

На рисунке выше приведена концептуальная схема темы оформления сайта на Drupal. Здесь мы имеем:

- Внешнюю обёртку, описанную в файле **html.tpl.php**, в котором задаются метаданные и верхний уровень документа HTML.
- Внутреннюю обёртку, или мастер-страницу, определяющую стандартный шаблон для всех страниц сайта и управляющую регионами.
- Регионы – области, в которых будет появляться (обычно динамичный) контент.

Схожий механизм темизации используется и внутри модулей – расширений для системы, отдельно он рассматриваться не будет.

2.3 Устройство ядра CMS Drupal и механизмы расширения

Рассмотрим устройство ядра Drupal. На диаграмме ниже приведены основные элементы системы, и, как можно заметить, почти все классы являются производными от **Entity** – класса, созданного для введения нового уровня абстракции с целью унифицировать создание сущностей разного типа и определения прав работы с ними.

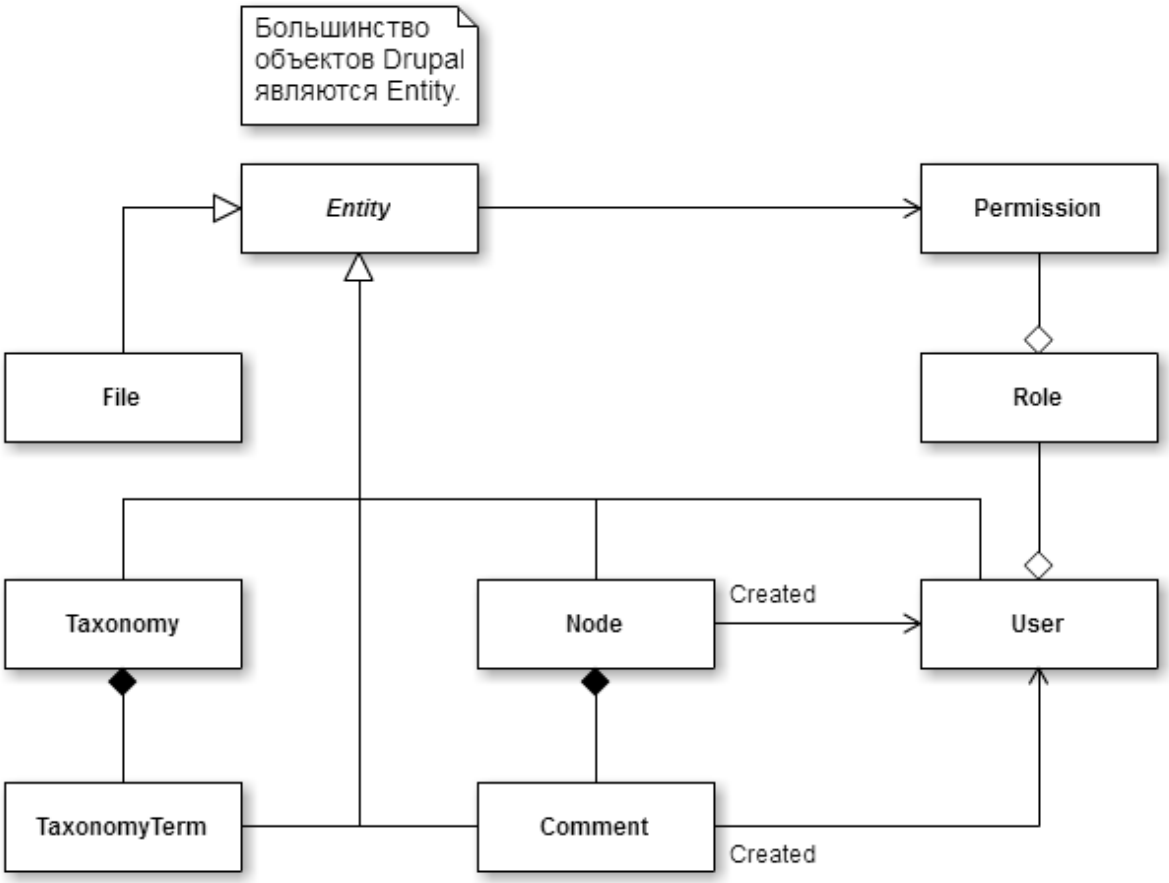


Рисунок 16 – Диаграмма классов ядра Drupal

Рассмотрим некоторые из представленных классов:

- **User.**
  Является производным от Entity и представляет собой пользователя системы.

- **Role.**
  Представляет собой роль пользователя в системе. По-умолчанию в системе есть 3 роли:
  - Зарегистрированный пользователь.
  - Модератор.
  - Администратор.
- **Permission**.
  Отвечает за права, доступные пользователю. Права выдаются пользователю не напрямую, а через связь с ролью, то есть, в явном виде права даются на роль, а у пользователя уже может быть набор ролей.
- **Node.**
  Отвечает за материалы на сайте – ноды.

На последнем классе предлагается остановиться поподробнее и рассмотреть его диаграмму классов [8], представленную ниже.

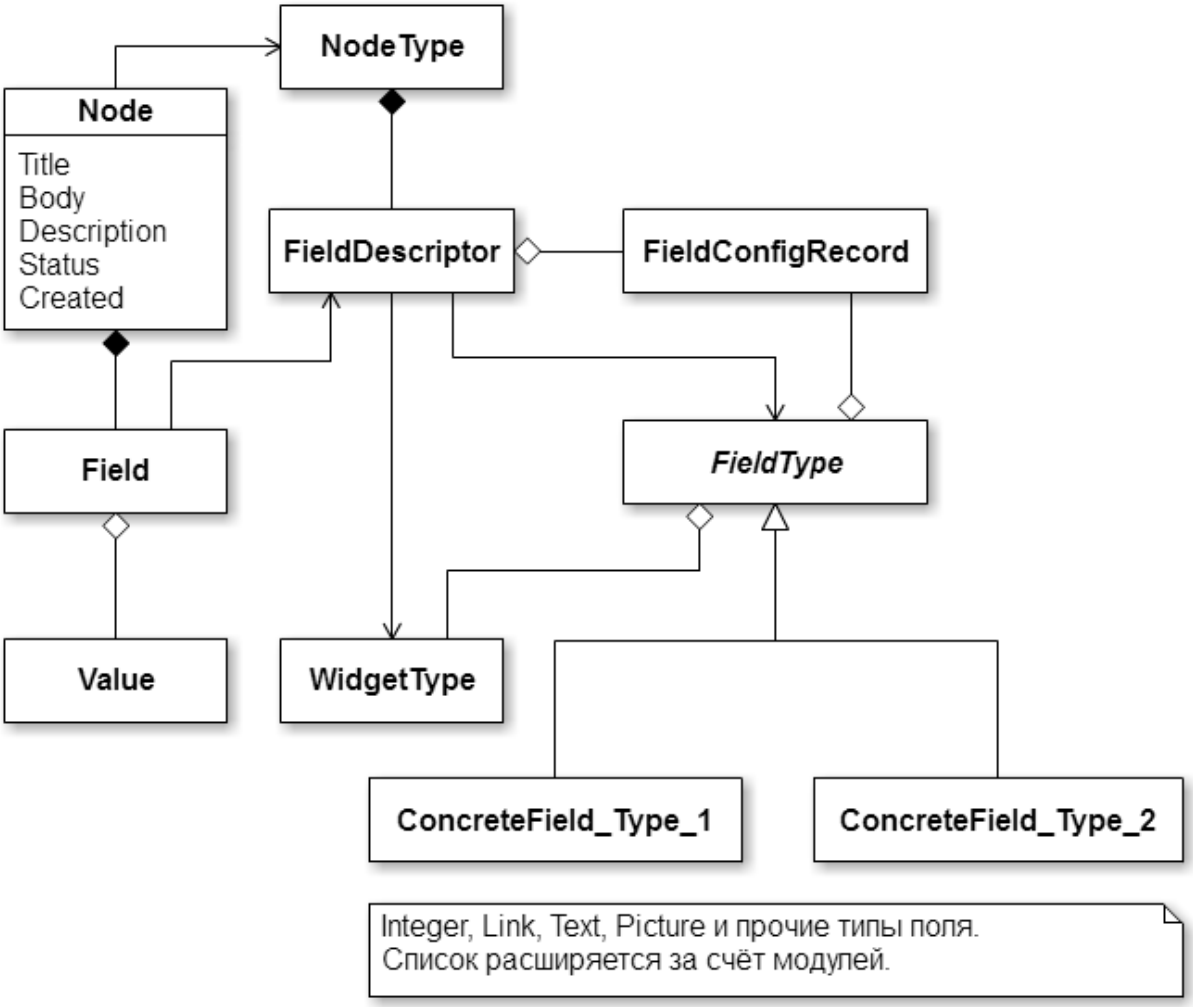


Рисунок 17 – Диаграмма классов ноды

Абсолютное большинство материалов, создаваемых на сайте, принято делать с использованием **Node**. Расширение ведётся путём создания собственных **NodeType** – типов материала, а также классов, реализующих абстрактный **FieldType** – тип поля.

Поле, в данном случае, это блок информации с определённым типом. Будь то целое число, короткий или длинный текст, изображение, или что-то более сложное и комплексное, например, некий граф, или включать в себя группу других полей.

Рассмотрим представленные классы:

- **Node**.
  Основной класс на диаграмме выше. Хранит базовую информацию о материалах (нодах).
- **NodeType**.
  Тип материала (ноды), сам по себе хранит описание и метаинформацию о типе материала, который описывается с помощью классов, связанных композицией. По сути, тип материала — это шаблон, по которому будут создаваться конкретные материалы.
- **FieldDescriptor**.
  Класс, описывающий поле в типе материала. С помощью него для типа материала задаётся список полей, доступный для заполнения в конкретных материалах.
- **FieldConfigRecord**.
  Класс, описывающий конфигурацию поля. Сюда же входят ограничения на поле (длина, его обязательность и прочее). Ограничения бывают как у конкретного описателя поля, так и у типа поля.
- **FieldType**.
  Класс, описывающий тип поля, абстрактный. Является точкой расширения.
- **WidgetType**.
  Класс, описывающий тип виджета. Виджет – визуальный инструмент, используемый при заполнении поля, к которому он привязан. В качестве примера можно указать два виджета для заполнения даты. В первом случае, это просто текст, где пользователь по шаблону вводит число, а во втором, это всплывающий календарь, где дата выбирается путём клика на соответствующее число.
- **ConcreteField_Type_1**
  Класс, описывающий конкретный тип поля.

- **Field**.
  Класс, описывающий конкретное поле конкретной ноды, привязан к описателю поля.
- **Value**.
  Класс, хранящий значение конкретного поля конкретной ноды.

Изучив устройство архитектуры Drupal, можно повторно изучить диаграмму предметной области и изменить её с учётом того факта, что известно окружение, в рамках которого система будет разрабатываться. На диаграмме ниже представлена диаграмма классов предметной области с учётом архитектуры Drupal.

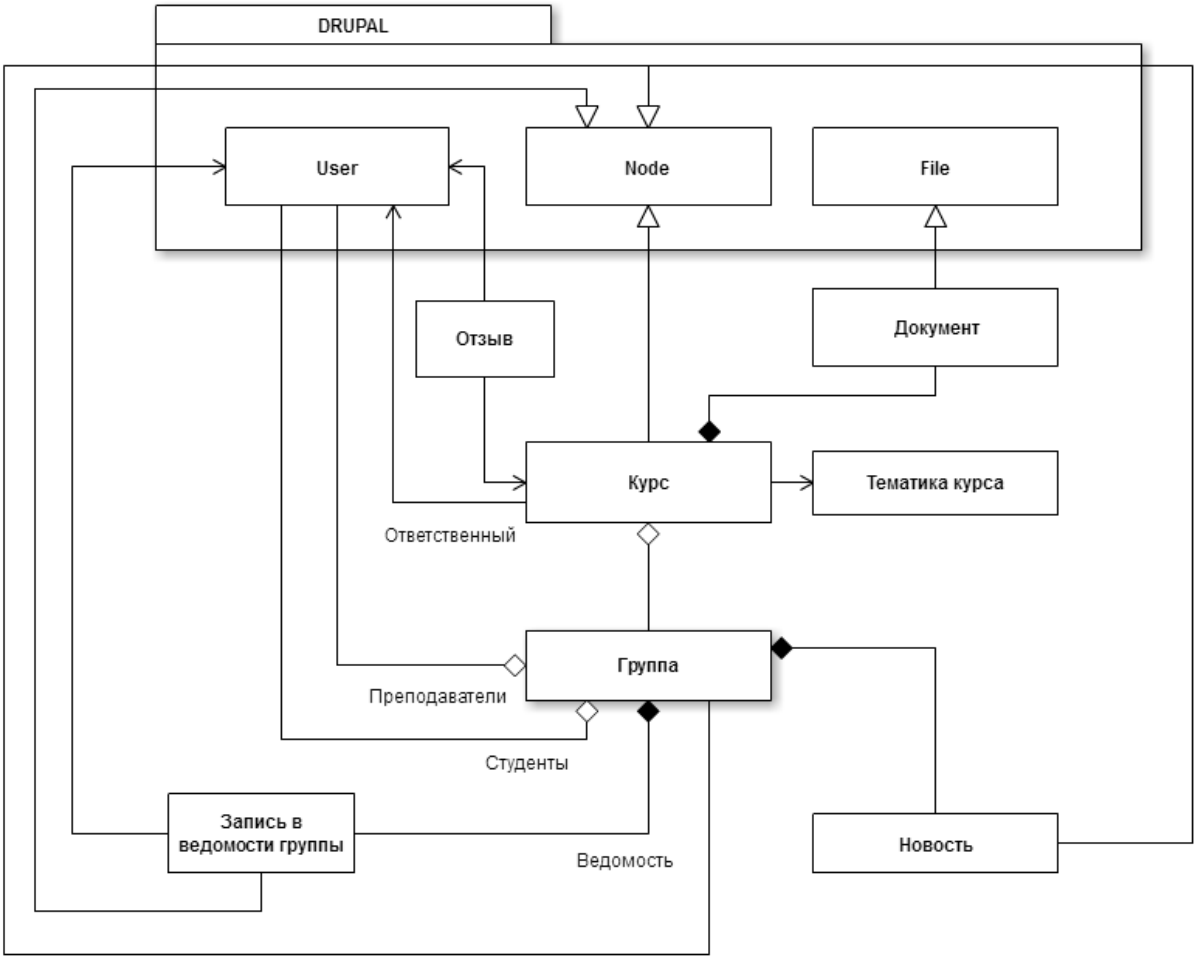


Рисунок 18 – Диаграмма классов разрабатываемой системы

Для расширения сайтов, реализованных в CMS Drupal принято использовать систему модулей. В случае с модулями, в Drupal, как и с темами, принято не изменять сторонние модули, если возникает необходимость внести правки в логику или отображение, а создавать свои модули, которые переопределят поведение расширяемых модулей и позволят изменяемым модулям продолжать получать обновления и оставаться в актуальном состоянии. Далее будет коротко рассмотрены принципы создания собственных модулей.

При разработке модулей так же, как и в случае с темами, используется механизм хуков, определение которых дано в разделе 2.3. При этом, основные файлы модуля следующие:

- **Modulename.info.**
  В данном файле описывается основная информация о модуле, такая как:
  - Имя модуля.
  - Описание модуля.
  - Версия модуля.
  - Имя пакета модуля – используется для группирования однотипных модулей в административной панели.
  - Список подключаемых файлов – необходим, если часть функций вынесены во вспомогательные файлы (например, с расширением .inc).
  - Требования к версии ядра.
  - Список зависимостей – список модулей, которые должны быть установлены и активированы.
- **Modulename.install**.
  В данном файле описываются функции, которые будут запущены при первом включении (установке) модуля, в том числе при его обновлении. Тут также используются хуки, так как именно они и будут вызваны в ходе установке. Вот список основных хуков:
  - **hook_schema.**
    В данной функции описывается схема базы данных в виде ассоциативного массива. При вызове данной функции отсутствующие таблицы будут созданы, а существующие изменены.
  - **hook_install.**
    В данной функции описываются все операции, которые должны быть выполнены при установке, вызывается после hook_schema. Обычно тут создаются типы материалов, создаются записи в БД, создаются и инициируются глобальные переменные сайта.

- **hook_update_N.**
  В данной функции описываются процедуры, выполняемые при обновлении от предыдущей версии модуля к следующей. Вместо N пишется номер обновления.

- **Modulename.module**.
  В данном файле описывается основной код модуля, именно тут реализуются необходимые хуки. Ниже приведён список основных хуков, которые встречаются чаще всего:
  - **hook_menu.**
    В данной функции описывается ассоциативный массив, задающий страницы по фиксированным адресам. При этом, страница может быть вида «pagename» – с фиксированным обработчиком конкретной страницы, так и «page/%» – обработчик вешается на группу страниц, удовлетворяющих описываемому шаблону.
    При этом, существуют варианты, при которых страница не отдаётся, а просто выполняется некоторая функция (callback).
  - **hook_theme.**
    В данной функции определяются функции темизации, которые могут быть использованы далее в модуле для оборачивания данных в HTML.
  - **hook_block_info.**
    В данной функции определяются блоки, создаваемые в модуле.
  - **hook_block_view.**
    В данной функции определяются правила вывода контента для блоков.

2.4 Интеграция с другими сервисами

В рамках данной работы также возникла необходимость провести интеграцию с другими информационными сервисами ТГУ. Для её реализации было использовано API, предоставляемое этими сервисами. Всё взаимодействие ведётся посредством HTTP-запросов. Далее приведён пример такого взаимодействия на примере диаграммы деятельности, иллюстрирующая процесс аутентификации пользователя через систему accounts.tsu.ru. Прочие взаимодействия направлены только на получение данных и практически не отличаются, в виду чего отдельно не демонстрируются.

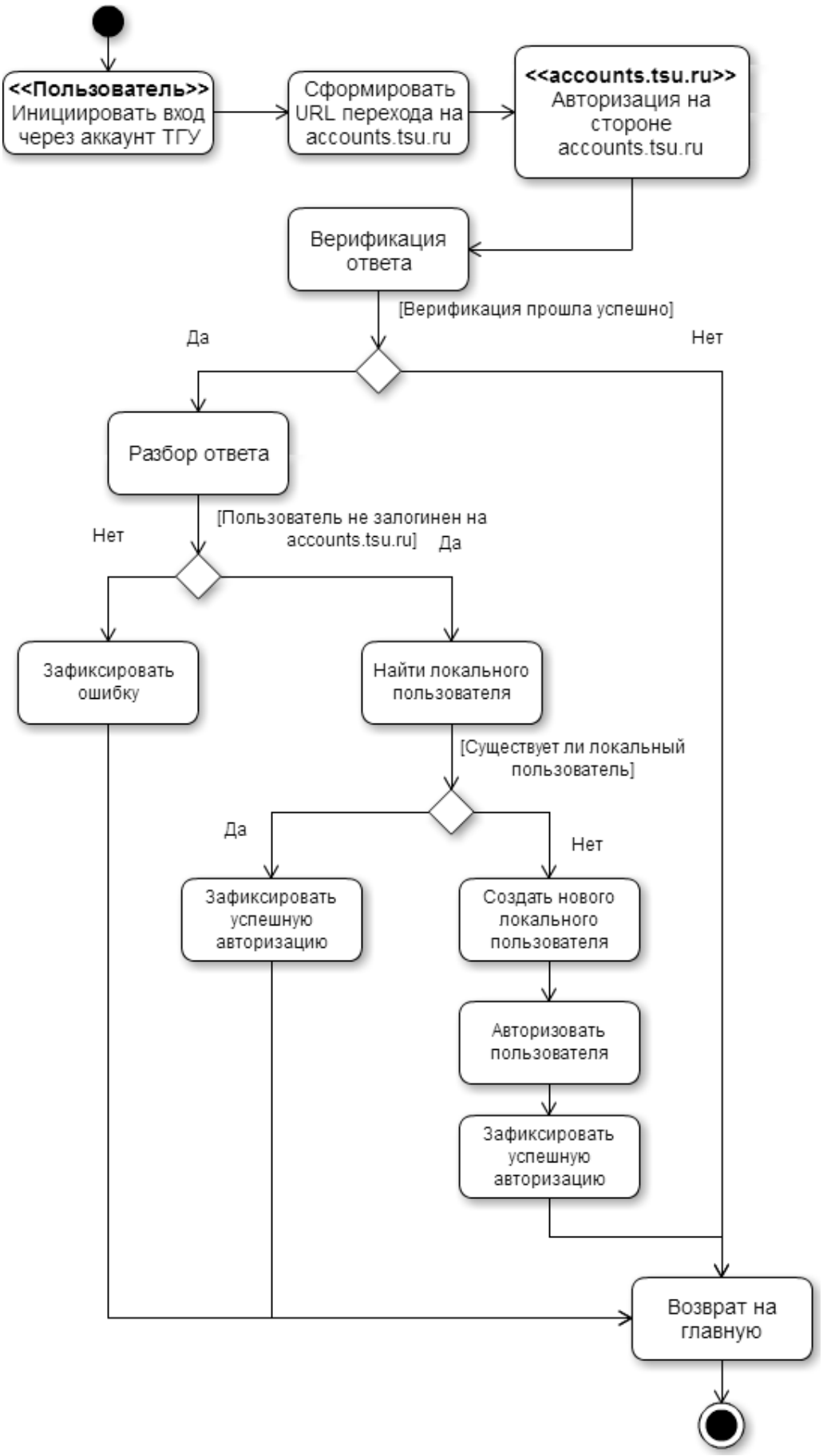


Рисунок 19 – Диаграмма деятельности для аутентификации через систему accounts.tsu.ru

3 Реализация.

3.1 Демонстрация разработанного функционала

При реализации системы был разработан модуль для Drupal, в рамках которого были реализованы классы, обозначенные на рисунке 18.

Для демонстрации разработанных механизмов, ниже будет рассмотрен вариант использования, представленный в таблице 4 – «Обновить содержание курса».

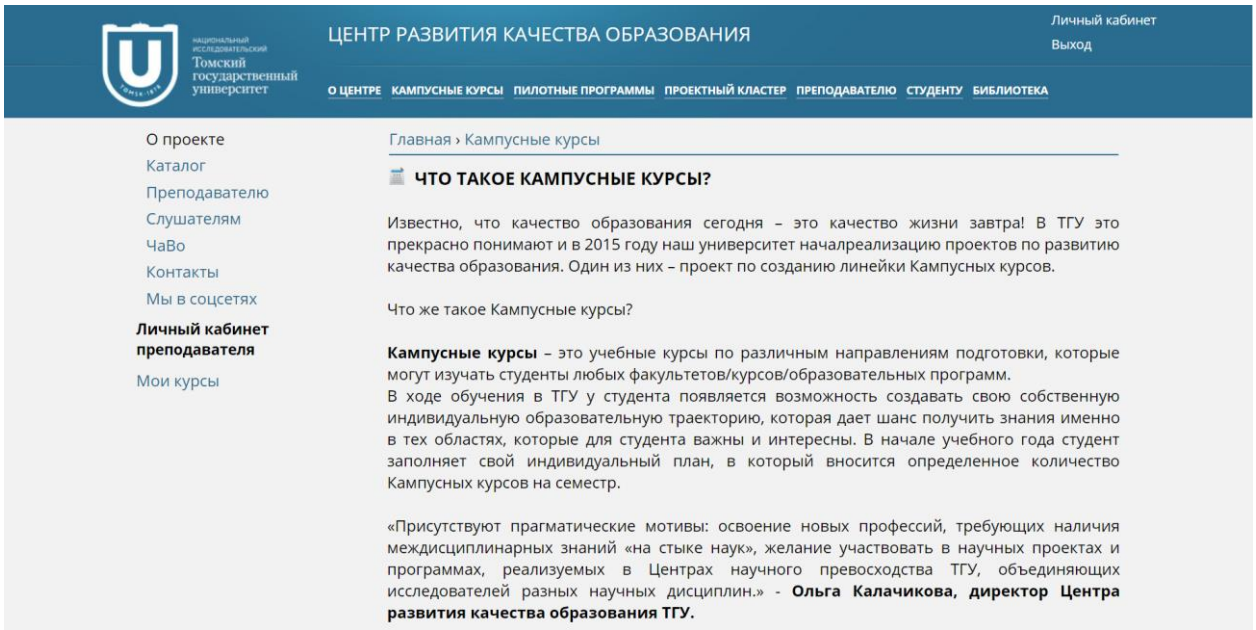


Рисунок 20 – Демонстрация архитектурно значимого варианта использования «Обновить содержание курса», страница «о проекте»

Шаг первый – перейти на страницу личного кабинета. Для этого пользователь нажимает на кнопку «личный кабинет», расположенную в верхнем правом углу сайта. В результате система переводит пользователя на страницу личного кабинета. Пример данной страницы приведён ниже.

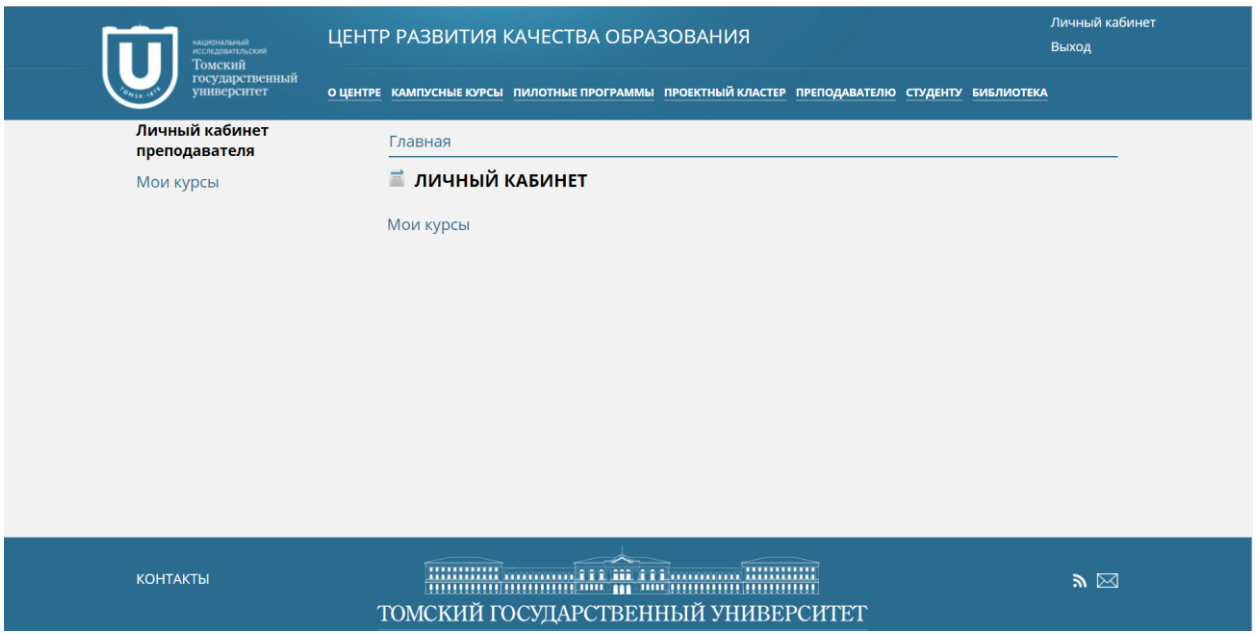


Рисунок 21 – Демонстрация архитектурно значимого варианта использования «Обновить содержание курса», страница личного кабинета

На странице личного кабинета у текущего пользователя есть только одно доступное действие – перейти на страницу курсов пользователя, нажав на любую из двух ссылок «мои курсы».

Шаг второй – перейти на страницу курсов пользователя. В результате система переводит пользователя на страницу с его курсами. В данном случае у текущего пользователя только один курс, который он ведёт, и он отображается на экране.

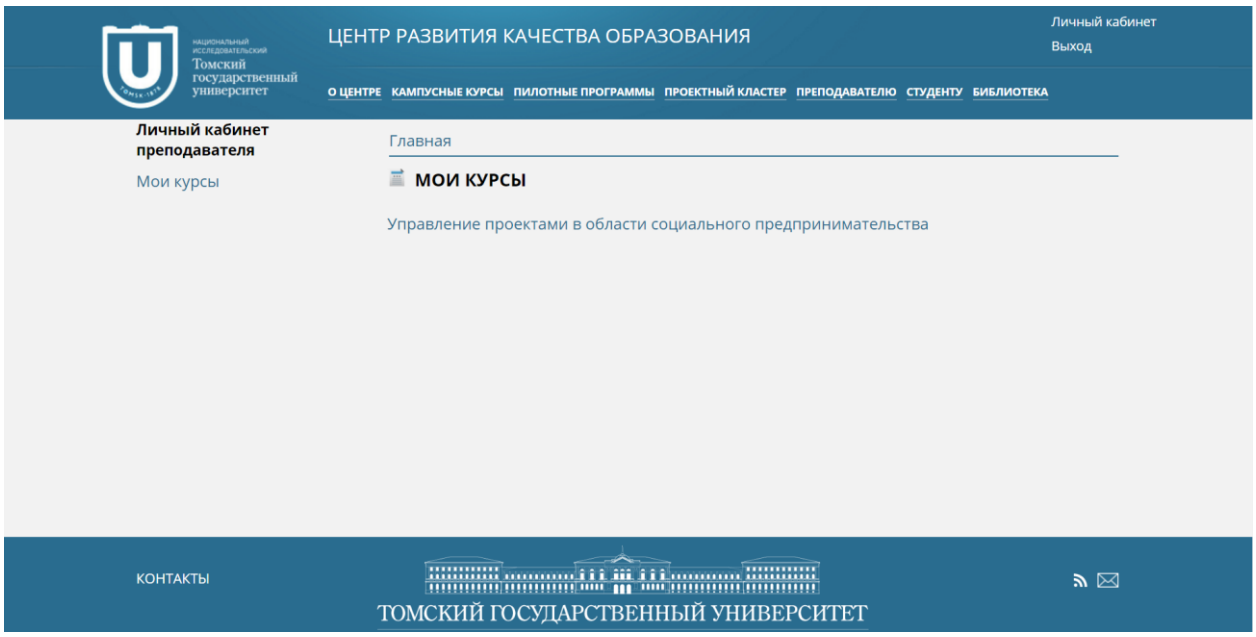


Рисунок 22 – Демонстрация архитектурно значимого варианта использования «Обновить содержание курса», страница «мои курсы»

Шаг третий – перейти на страницу курса. Для этого необходимо выбрать курс, который будет отредактирован.

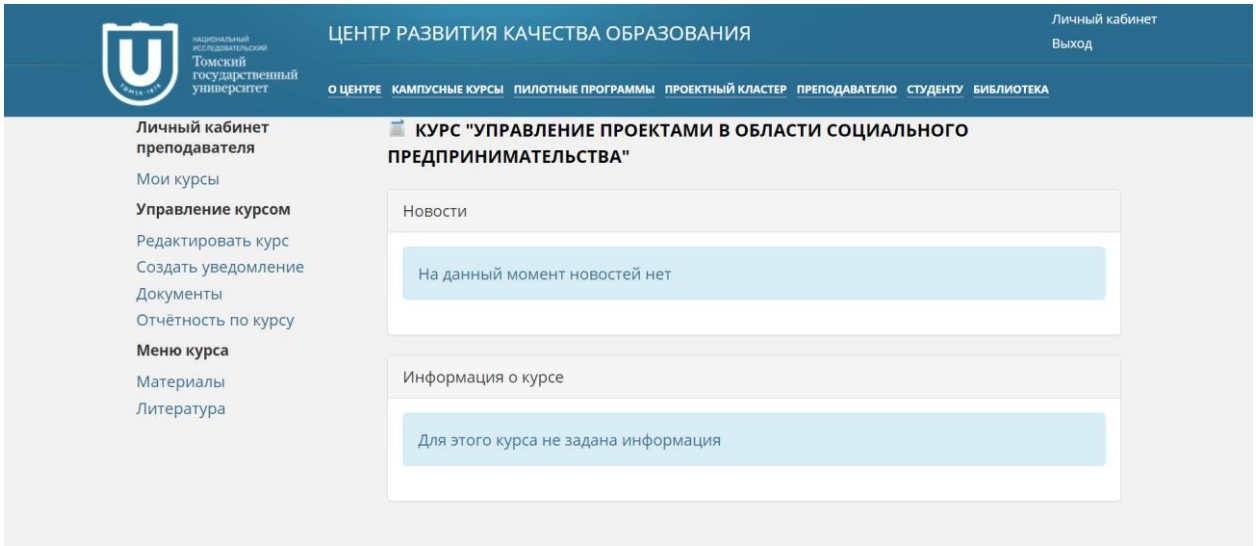


Рисунок 23 – Демонстрация архитектурно значимого варианта использования «Обновить содержание курса», внутренняя страница курса

На рисунке выше изображена внутренняя страница курса, на которой у пользователя с ролью «преподаватель» есть боковое административное меню, позволяющее пользователю в оперативном режиме открывать необходимые разделы.

Данную страницу (внутреннюю страницу курса) видят только пользователи, указанные преподавателями у курса, а также студенты, числящиеся участниками курса.

В центральной же части страницы есть два функциональных раздела – «новости» и «Информация о курсе».

Раздел «новости» заполняется новостями, созданными пользователями «преподавателями» в рамках данного курса.

Раздел «информация о курсе» выводится в том случае, если преподаватель заполнил аннотацию к курсу.

Шаг четвёртый – выбрать режим «редактировать курс». Для этого на экране, изображённом на рисунке 22 необходимо выбрать соответствующую ссылку в боковом административном меню.

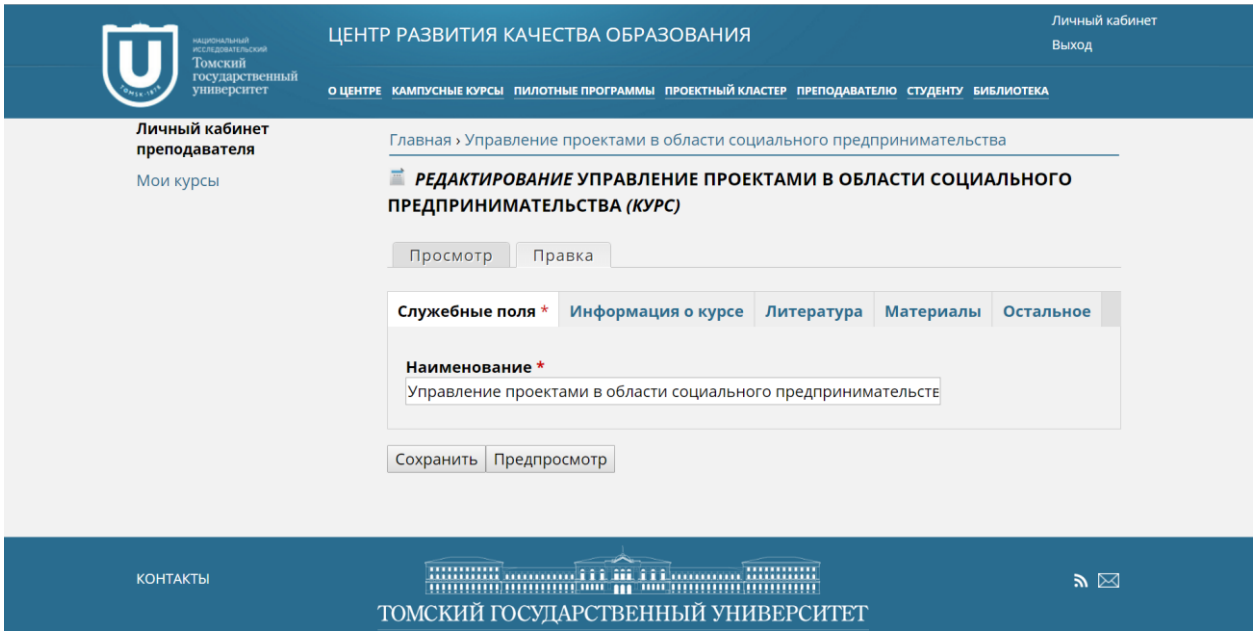


Рисунок 24 – Демонстрация архитектурно значимого варианта использования «Обновить содержание курса», страница редактирования курса

На странице редактирования курса пользователю доступны 5 вкладок для редактирования доступной информации. В зависимости от роли пользователя, число отображаемых и доступных для редактирования полей может изменяться. Например, пользователю с ролью «преподаватель» на вкладке «служебные поля» доступно только редактирование наименования курса, в то время как у пользователя с ролью «администратор» есть гораздо больше доступных для редактирования опций (в том числе открытость курса для записи и прочее).

На последующих изображениях приведены примеры редактирования информации о курсе, доступной пользователю с ролью «преподаватель».

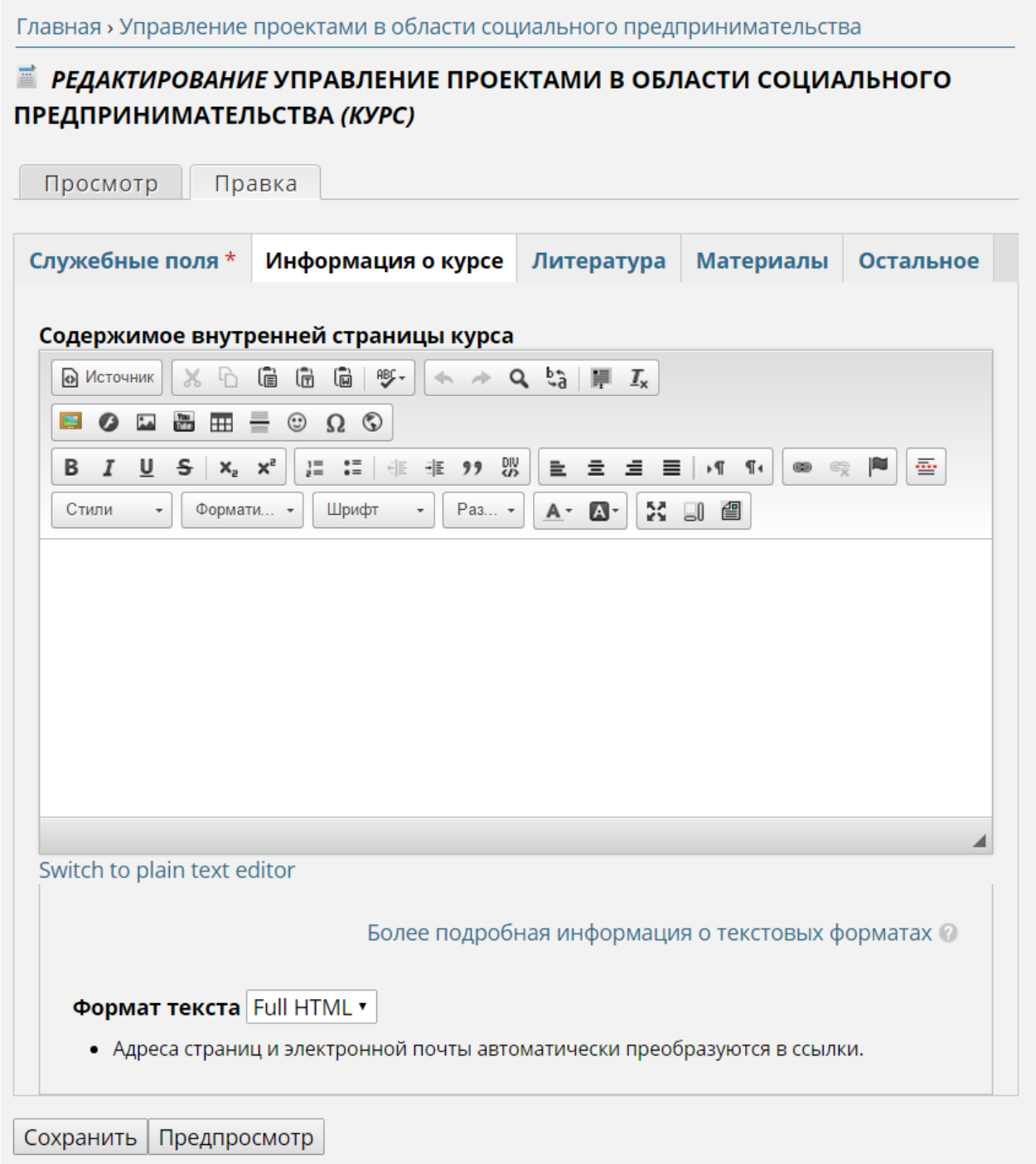


Рисунок 25 – Демонстрация архитектурно значимого варианта использования «Обновить содержание курса», страница редактирования курса, информация о курсе

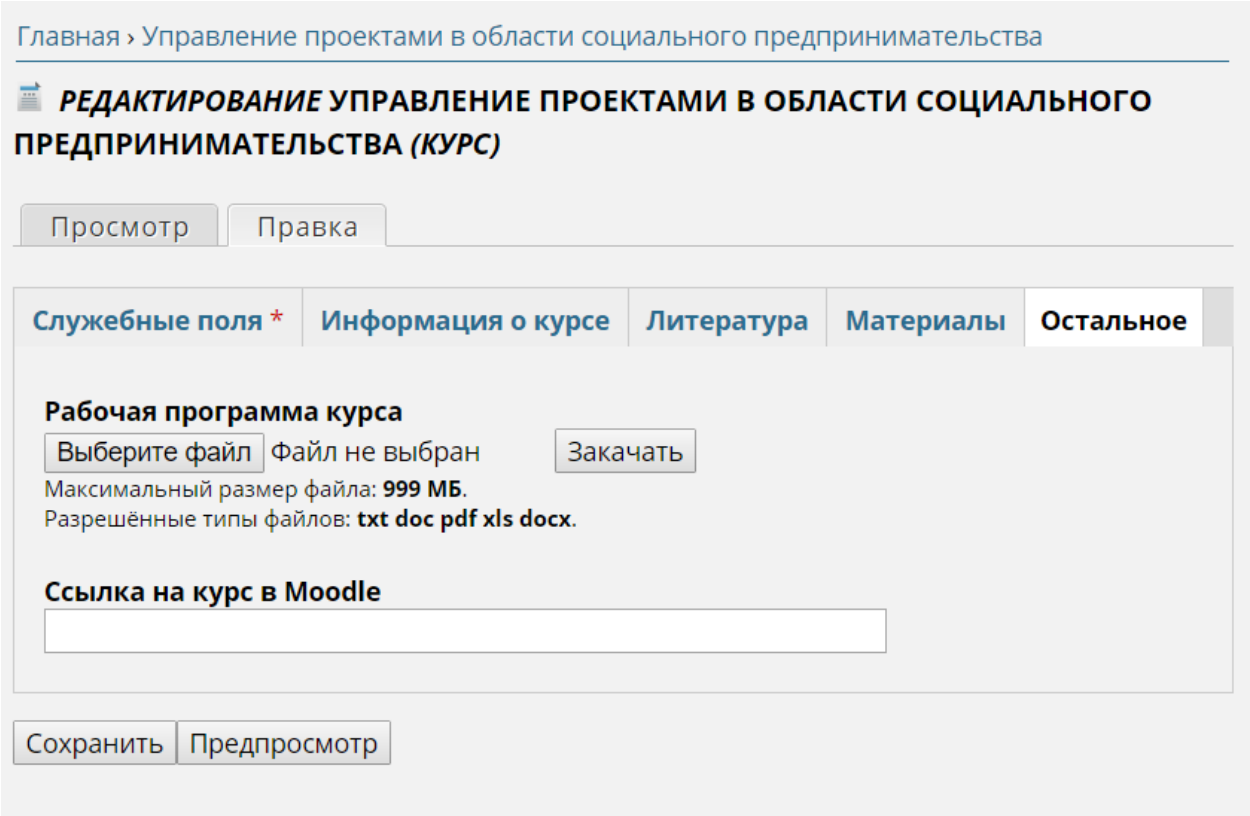


Рисунок 26 – Демонстрация архитектурно значимого варианта использования «Обновить содержание курса», страница редактирования курса, остальное

По завершению редактирования информации, пользователь инициируют сохранение нажатием на кнопку «сохранить». После этого, система вызывает процесс верификации введённой информации и, в случае его сбоя, возвращает пользователя на страницу редактирования, подсвечивая поля, где информация была введена некорректно.

3.2 Внедрение

Для внедрения разработанного решения на сервера ТГУ были выполнены следующие действия:

1. Разработанное решение протестировано на предмет корректной работы.
2. Разработанные модули и темы скопированы в соответствующие каталоги расширяемого сайта на серверах ТГУ.
3. На расширяемом сайте выполнена процедура обновления посредством вызова /**update.php**.
4. После обновления установлены и активированы необходимые модули.
5. Произведена необходимая конфигурация модулей и сайта.
6. Проведено повторное тестирование разработанного решения на конечном сервере.

ЗАКЛЮЧЕНИЕ

В рамках данной работы решены все поставленные задачи:

- Собраны и формализованы требования к разработанной системе.
- Спроектирована разработанная система.
- Разработана информационная система управления кампусными курсами.
- Протестирована и внедрена разработанная система.

Таким образом, цель — разработать и внедрить систему управления кампусными курсами ТГУ — достигнута, система управления кампусными курсами запущена и на момент написания данной работы создано свыше 150 кампусных курсов.

Разработанный ресурс доступен по адресу http://cdeq.tsu.ru.

ЛИТЕРАТУРА

1. Drupal 7 Deconstructed [Электронный ресурс] // URL: http://www.drupaldeconstructed.com/content/ (дата обращения: 15.01.2016).
2. Drupal API [Электронный ресурс] // URL: https://api.drupal.org/api/drupal (дата обращения: 25.03.2015).
3. MVC vs. PAC [Электронный ресурс] // URL: http://www.garfieldtech.com/blog/mvc-vs-pac (дата обращения: 20.02.2016).
4. Presentation–abstraction–control [Электронный ресурс] // URL: https://en.wikipedia.org/wiki/Presentation%E2%80%93abstraction%E2%80%93control (дата обращения: 20.02.2016).
5. Cockburn A. Writing Effective Use Cases. 1st ed. Addison-Wesley Professional, 2000. 304 pp.
6. Фаулер М. Архитектура корпоративных систем. Москва: Издательский дом "Вильямс", 2006. 514-518 pp.
7. Гамма Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма [и др.]. – СПб: Питер, 2001. – 368 с.
8. Ларман К. Применение UML и шаблонов проектирования. / К. Ларман – Издательский дом «Вильямс», 2004. 620 с.
9. Мелансон Б. Профессиональная разработка сайтов на Drupal 7 : пер. с англ. / Б. Мелансон [и др.]. – СПб: Питер, 2013. - 687 с.
10. Справочник по CSS [Электронный ресурс] // URL: http://htmlbook.ru/css (дата обращения: 01.05.2014).
11. Справочник по HTML [Электронный ресурс] // URL: http://htmlbook.ru/html (дата обращения: 01.05.2015).
12. Справочник по PHP [Электронный ресурс] // URL: http://www.php.su (дата обращения: 25.04.2015).

Укажите год публикации: 2016

Выберите коллекции

- [x] Все
- [x] Рефераты
- [x] Авторефераты
- [x] Иностранные конференции
- [x] Википедия
- [x] Российские конференции
- [x] Иностранные журналы
- [x] Российские журналы
- [x] Энциклопедии
- [x] Англоязычная википедия

Анализировать

Обработан файл:
Отчёт_актуально.docx.

Год публикации: 2016.

Оценка оригинальности документа - 100.0%

Процент некорректных заимствований - 0.0%

Просмотр заимствований в документе

Время выполнения: 28 с.

Заимствования отсутствуют