**СОДЕРЖАНИЕ**

[Введение](#_Toc452938673) 6

[1 Теоретические основы онлайн обучения с использованием информационных технологий](#_Toc452938674) 8

[1.1 Онлайн обучение: понятие и виды](#_Toc452938675) 8

[1.2 Обзор существующих аналогов веб-приложений для онлайн обучения](#_Toc452938676) 15

[2 Разработка требований и функциональной модели](#_Toc452938679) 23

2[.1 Выбор языка программирования, платформ и средств разработки ПС](#_Toc452938677) 23

[2.2 Разработка функциональной модели](#_Toc452938682) 27

[2.3 Требования к программному средству](#_Toc452938681) 34

[3 Проектирование и реализация программного средства](#_Toc452938684) 37

[3.1 Общая структура программного средства](#_Toc452938685) 37

[3.2 Разработка модели базы данных](#_Toc452938685) 38

[3.3 Программная реализация](#_Toc452938687) 42

[3.4 Тестирование](#_Toc452938688) 46

[4 Руководство пользователя](#_Toc452938692) 53

[4.1 Пользовательская роль «Гость»](#_Toc452938692) 53

[4.2 Пользовательская роль «Слушатель»](#_Toc452938692) 54

[4.3 Пользовательская роль «Лектор»](#_Toc452938692) 55

[4.4 Пользовательская роль «Администратор»](#_Toc452938692) 57

[5 Технико-экономическое обоснование разработки и использования веб-приложения](#_Toc452938699) 59

[Заключение](#_Toc389183055) 73

[Список использованных источников](#_Toc389183056) 74

[Приложение А Исходный код веб-приложения](#_Toc452938707) 75

# Введение

Современный университет – многофункциональная многоуровневая система, интегрирующая многочисленные функции: ведение образовательной, воспитательной, научной, производственной, издательской и прочей деятельности. Современное образование развивается в контексте общемировых тенденций глобализации, информатизации, стремительного развития образовательных технологий, их приближения к основным потребителям, создания открытых образовательных ресурсов.   
Реалии современного образовательного процесса побуждают преподавателей к выходу за рамки традиционной схемы обучения, к максимально широкому использованию технологий смешанного обучения, к дополнению проверенных педагогических методов современными онлайн-технологиями.

Использование современных средств обучения, предоставляемых большинством систем управления обучением, позволяет снять большинство трудностей онлайн-обучения, приблизив его к традиционному ­аудиторному, сделать процесс обучения более эффективным, удобным, открыть образование различным категориям обучающихся, которые в силу определенных причин не могут находиться в аудитории (люди с ограниченными физическими возможностями, работающие люди и пр.), расширить границы обучения с размеров одной аудитории до целого мира.

Развитие современных средств обучения, широкая доступность образовательных ресурсов, технологизация и информатизация общества, практически повсеместное и постоянное наличие у студентов мобильных устройств (смартфоны, планшеты, ноутбуки) лишают преподавателя выбора – ему предстоит либо трансформироваться вместе со своими студентами, обществом, соответствовать им, либо безнадежно от них отстать. Такая же участь ждет и университет: без развития и внедрения современных электронных форм, средств и методов в процесс обучения невозможно представить его динамичное развитие, а промедление приведет к нарастанию разрыва между лидерами и отстающими.

Альтернатива развития либо онлайн-обучения, либо оптимизации сложившегося преподавания, таким образом, неуклонно трансформируется в насущную необходимость одновременного развития обоих направлений, т. е. поступательной трансформации существующих образовательных практик и обеспечения присутствия на мировом рынке открытых образовательных ресурсов.

Объектом исследования и разработки является веб-приложения для онлайн обучения с поддержкой мультимедийного контента.

Целью проекта является разработка программного комплекса, который является веб-приложением для онлайн обучения с поддержкой мультимедийного контента.

Основные задачи дипломного проекта:

* изучить теоретические основы онлайн обучения с использованием информационных технологий;
* определить основные требования к веб-приложению;
* создать функциональную модель разрабатываемого программного средства;
* спроектировать и разработать веб-приложение для онлайн обучения с поддержкой мультимедийного контента;
* рассчитать технико-экономические показатели разработки и внедрения веб-приложения.

1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ОНЛАЙН ОБУЧЕНИЯ С ИСПОЛЬЗОВАНИЕМ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
   1. Онлайн обучение: понятие и виды

Системы высшего образования во всем мире сегодня широко используют современные информационно-коммуникационные технологии (ИКТ), которые позволяют не только объединить традиционные инструменты обучения, но и значительно расширить их перечень, оказывая существенное влияние на информационную культуру в образовательной среде вуза. Все более востребованным способом получения новых знаний в мире становится электронное обучение. Несмотря на значительное число исследований по использованию ИКТ в образовании, многие аспекты использования электронного обучения разработаны недостаточно. Внедрение электронного обучения в образовательную среду университета требует его теоретического осмысления, создания научно обоснованной системы подготовки всех участников образовательного процесса.

В настоящее время существуют различные трактовки понятия «онлайн обучение». Дополнительную сложность создает тот фактор, что такие понятия, как «элекронное обучение», «веб-ориентированное обучение», «онлайн обучение» и «дистанционное обучение», часто взаимозаменяются.

В представленной работе эти термины будут рассмотрены более подробно для выявления их ключевых различий и уточнения формулировки понятия «онлайн обучения».

Значения технических терминов могут быть определены разными способами. Определения «электронное обучение» и «веб-ориентированное обучение» приобрели свое содержание благодаря представлениям об электронной почте, всемирной паутине и обучении как таковом. В качестве примеров образования новых терминов можно привести также появление таких понятий, как «онлайн обучение» и «дистанционное обучение».

Толкования терминов «дистанционное обучение» и «дистанционное образование» были предложены Д. Киганом, Д.Р. Гэррисоном и Д. Шейлом. Б.Х. Хан и Б. Холл проводили параллели между веб-ориентированным обучением и технологиями, лежащими в основе работы веб-браузера. В работах М. Розенберга, М. Дрисколла, У. Хортона термин «электронное обучение» ассоциируется с термином «веб-ориентированное обучение». Р.К. Щенк соотносит электронное обучение с образовательной деятельностью, включающей в себя использование компьютерных сетей, и разводит понятия электронного и дистанционного обучения. Концепция онлайн обучения предшествует появлению всемирной паутины, однако публикации Л. Малопински, Дж. Киркли, Р. Штайна, Т. Даффи, Р.К. Щенка и американской некоммерческой общественной службы телевизионного вещания на тему онлайн обучения ссылаются на материалы, поставляемые посредством Интернета и Интранета.

Определения, представленные в технической литературе, могут разниться с общепринятым употреблением этих терминов ввиду чрезмерного обобщения либо неоправданного сужения сущности понятий. Иногда под электронным обучением подразумевается любая форма обучения при помощи электронных устройств, включая обучение, основанное на применении вычислительной техники. Некоторые авторы сводят представление о веб-ориентированном обучении к использованию учебных материалов, выложенных в сети Интернет, при этом исключая те же самые материалы, но уже поставляемые на CD-дисках.

Нижепредставленный подход к определению рассматриваемых терминов включает применение двух комплементарных методов: на основе индивидуальных значений и значений относящихся к ним концепций. Например, для толкования идеи онлайн обучения рассматриваются значения слов «онлайн» и «обучение», а также значения концепций, относящихся к этой сфере, таких как «онлайн-документация», «онлайн- сервис» и «онлайн-помощь». Также в рамках этого подхода изучается экспертная информация об использовании указанной терминологии и предложенных определений. Таким образом, были получены ключевые различия и частичные совпадения в трактовке терминов. Определение электронного обучения появилось из концепции электронной почты. Электронную почту описывают как процесс передачи индивидуальных сообщений посредством компьютеров и компьютерных сетей. В данном случае под компьютерными сетями понимаются Интернет и Интранет. Электронное обучение соотносится с деятельностью, включающей одновременное использование компьютеров и сетей интерактивной диалоговой связи. Компьютер и компьютерная сеть должны быть значительными элементами образовательного процесса [1, с. 145].

Веб-ориентированное обучениесвязано с применением учебных материалов в формате веб, поставляемых через веб-браузер, включая материалы на CD-дисках и других медианосителях. Возможность скачивания материалов курса компьютеризованной профессиональной подготовки с веб-сайта не является веб-ориентированным обучением, так как обучающимся не предоставляется контент в формате веб. Просмотр учебных материалов в формате веб является ключевой особенностью данной формы обучения. Онлайн-обучение основано на эксплуатации контента, всегда доступного на компьютере. В большинстве англоязычных словарей [2, 3] в определение понятия «онлайн» включается использование компьютеров и компьютерных сетей, преимущественно сети Интернет. Так, например, в группе Кембриджских словарей под термином «онлайн» подразумевают продукты, услуги и информацию, которые могут быть приобретены или использованы посредством сети Интернет. Однако, если про эксплуатацию компьютерных сетей говорится не во всех определениях данного понятия, то применение компьютера является обязательным атрибутом каждого из них. Так, в самом популярном англоязычном онлайн-словаре «онлайн» означает доступный или функционирующий посредством компьютера [4].

Веб-ориентированное обучение и необходимое программное обеспечение, доступное из основного приложения, также может носить название онлайн-обучения. Однако поиск или открытие отдельного приложения для того, чтобы получить доступ к учебным материалам, не оценивается как онлайн-обучение, так как материалы не находятся в открытом доступе.

Дистанционное образование представляет собой прогрессивный вид обучения, позволяющий объединить вместе в процессе обучения территориально удаленных преподавателя, учебный материал и обучающегося.

Надо отметить, что термин «дистанционное образование» получил значительноменьшее распространение по сравнению с термином «дистанционное обучение». Это в значительной степени объясняется тем, что распространение технологий дистанционного обучения происходит значительно быстрее становления законодательно иорганизационно обусловленной формы дистанционного образования [5].

В конце XX – начале XXI в. Появились усовершенствованные варианты дистанционного образования, включая сочетание программ корреспондентского обучения и онлайн-обучения. Программы корреспондентского обучения предполагают работу с учебниками и рабочими тетрадями, в рамках которой обучающийся может самостоятельно выбирать скорость изучения материала. Требованием некоторых учебных заведений может являться отправление по почте выполненных заданий, но к настоящему моменту обучающиеся по большей части могут представлять на рассмотрение свои работы, сдавать экзамены, просматривать свои оценки в режиме онлайн. В Интернете дистанционное обучение включает взаимодействие преподавателя и обучающихся, а также обучающихся между собой на расстоянии и делает возможным своевременное предоставление ответов преподавателя на вопросы обучающихся. Типичное дистанционное обучение в этом контексте включает в себя: интернеттрансляции в режиме реального времени, видеоконференции, чаты, запланированные онлайн-дискуссии и даже курсы или обсуждения, происходящие посредством использования сервиса электронной почты.

Позднее появилось электронное обучение, которое является разновидностью дистанционного обучения. К его достоинствам относится возможность построения индивидуальной траектории обучения, доступ к образовательным ресурсам в любое время и в любом месте (при наличии доступа к сети Интернет). Многие образовательные учреждения обеспечивают своих слушателей онлайн-версиями учебников и учебных пособий, так же как и другими учебными материалами, и даже виртуальными библиотеками. Обучающиеся имеют доступ к асинхронным электронным доскам объявлений и форумам, чатам в режиме реального времени, записанным видеолекциям и другим виртуальным возможностям взаимодействия с преподавателями и сокурсниками. Некоторые сессии в режиме реального времени требуют виртуального посещения в определенные дни и время, и задания нужно сдавать в строго обозначенные сроки, но все остальное доступно в любое время, что позволяет более гибко планировать свое рабочее и учебное время [1].

В таблице 1.1 приведены понятия всех вышерассмотренных форм обучения, их признаки, составляющие понятия термины, а также концепции, близкие по значению к рассматриваемым понятиям.

Таблица 1.1 – Понятия электронного, дистанционного, веб-ориентированного и онлайн-обучения

|  |  |  |  |
| --- | --- | --- | --- |
| Понятие | Составляющие термины и их значения | Концепции, близкие по  значению | Признаки |
| 1 | 2 | 3 | 4 |
| Онлайн  обучение  (online learning) | Онлайн, обучение  Онлайн – осуществляемый в режиме реального времени через Интернет; предоставляющий услуги через Интернет.  Первоначально использовалось только в  отношении коммуникационного оборудования для указания на режим связи, типичным значением могло быть «не вешая трубку», то есть за один телефонный звонок, в режиме реального времени | Онлайн-документация,  онлайн-сервис, онлайн-помощь.  Онлайн-помощь – это тематическая справочная  информация, доставляемая пользователю  посредством программного обеспечения | Использование компьютера;  Использование всегда  доступного контента  (в Интернете, всемирной паутине, на CD-  диске или жестком диске  компьютера) |
| Веб-ориенти-  рованное обуче-  ние (web-based  learning) | Веб, веб-технологии, обучение  Веб – «Всемирная паутина», т. е. глобальная гипертекстовая система для поиска и  использования ресурсов сети Интернет.  Предоставляет доступ к базам данных, возможность пользования электронной  почтой, проведения видеоконференций и  другие информационные услуги.  применением сети Интернет | Веб-дизайн, веб-хостинг,  веб-страница, веб-браузер, веб-приложение | Использование учебных  материалов в формате  веб (гипертекст, гиперс-  сылки) |

Продолжение таблицы 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Электронное  обучение (electronic  learning) | Электронный, обучение  Электронный – основанный на использовании компьютера, компьютерных сетей | Электронное правительство, электронная  экономика, электронная  коммерция, электронный  бизнес, электронная  почта | Использование компьютерных сетей,  электронных носителей;  Доступ к образовательным ресурсам в любое  время и в любом месте |
| Дистанционное  обучение (distance  learning) | Дистанционный, обучение  Дистанционный – на отдаленности, на  расстоянии |  | Наличие обратной  связи;  Удаленность участников процесса обучения;  Использование компьютеров и компьютерных сетей не является  обязательным |

Процесс обучения представляет собой взаимодействие педагога, обучаемого и средств обучения. Возможности современных компьютерных средств и информационных технологий позволяют возложить на средства обучения часть функций преподавателя и часть функций обучаемого, принятых в классической форме обучения. Информационные технологии в онлайн обучении играют роль инструментов, которые:

* обеспечивают учащимся удаленный доступ не только к учебному материалу, но и к большому количеству справочной информации, в дополнительной, сопровождающей форме;
* предоставляют учащимся средства общения с онлайн репетитором, а также между собой; сегодня имеются технические возможности для того, чтобы учащийся, находящийся на большом расстоянии от учебного центра (школы, вуза, колледжа, лицея), прослушал и просмотрел лекцию, принял участие в видеоконференции или получил консультацию, выполнил компьютерный лабораторный эксперимент и т. п.;
* осуществляют управление и контроль за процессом обучения; обучаемый должен убедиться, прежде всего сам в том, что разобрался в изучаемом учебном материале, понял его, запомнил основные положения, научился применять их на практике для решения практических задач. С другой стороны, активная роль онлайн преподавателя не менее существенна, поскольку его задача не только убедиться в знаниях подопечного, но и — как при очном обучении – принять решение по корректировке программы обучения с тем, чтобы добиться наилучшего усвоения пройденного материала;
* предоставляют возможность создания эффективных тренажеров, средств визуализации, максимальное использование различных способов представления информации: текста, графики, видео, звукового сопровождения, анимации, т. е. то, что получило название "мультимедиа" [6].

Ключевым элементом построения обучения с использованием интернет-технологий является обеспечение удаленного доступа к учебному контенту. Широкое распространение получают такие разновидности онлайн обучения как виртуальная школа, дистанционные курсы и.т.д. Ниже автором более подробно рассмотрены эти виды онлайн обучения.

Виртуальная школа – образовательное учреждение, в котором педагогический процесс и обучение осуществляются через Интернет. Материалы по учебным курсам в виртуальной школе представлены в электронном виде и выкладываются на веб-сайте таким образом, чтобы прошедшие авторизацию обучающиеся могли ими пользоваться. Эти материалы обычно включают в себя тексты лекций по предмету, интерактивные тесты и тренажеры, словари и т. д. Ознакомившись с материалами виртуального урока, обучающийся выполняет ряд заданий, которые автоматически проверяются системой, с выставлением оценки. Обучающийся может вступать во взаимодействие с сетевыми преподавателями, консультируясь по предметам. Сетевые преподаватели могут также осуществлять контроль и оценку знаний обучающийся, общаясь с ним по электронной почте, по телефону, в форуме или при помощи иных технических средств связи. Аттестация обучающихся по всему курсу обычно осуществляется в форме экзамена (очного или заочного) [6].

Дистанционный курс – особая, основанная на использование современных информационных технологий, форма представления содержания учебного курса. Дистанционный курс является основным элементом построения обучения с использованием технологий дистанционного обучения

При помощи Интернет - технологий возможны следующие формы занятий:

а) чат-занятия – учебные занятия, осуществляемые с использованием чат-технологий. Чат-занятия проводятся синхронно, то есть все участники имеют одновременный доступ к чату. В рамках многих дистанционных учебных заведений действует чат-школа, в которой с помощью чат-кабинетов организуется взаимодействие педагогов и учеников;

б) веб-занятия – дистанционные уроки, конференции, семинары, деловые игры, лабораторные работы, практикумы и другие формы учебных занятий, проводимых с помощью средств телекоммуникаций и других возможностей «Всемирной паутины». Для веб-занятий используются специализированные образовательные веб-форумы — форма работы пользователей по определённой теме или проблеме с помощью записей, оставляемых на одном из сайтов с установленной на нем соответствующей программой.

От чат-занятий веб-форумы отличаются возможностью более длительной (многодневной) работы и асинхронным характером взаимодействия учеников и педагогов;

в) телеконференции – проводятся, как правило, на основе списков рассылки с использованием электронной почты. Для учебных телеконференций характерно достижение образовательных задач. Также существуют формы дистанционного обучения, при котором учебные материалы высылаются почтой в регионы;

г) онлайн-семинар – разновидность веб-конференции, проведение онлайн-встреч или презентаций через Интернет в режиме реального времени. Во время веб-конференции каждый из участников находится у своего компьютера, а связь между ними поддерживается через Интернет посредством загружаемого приложения, установленного на компьютере каждого участника, или через веб-приложение [6].

Видеолекция в последнее время заменила традиционные лекции и стала неотъемлемым видом дистанционного обучения. Инновационные виды обучения предполагают использование видеолекций, хранящихся на индивидуальном электронном носителе или веб-сервере.

Основная цель видеолекции заключается в оказании на учащегося наибольшего эмоционального воздействия, плодотворном вовлечении его в учебный процесс.

Преимущество видеолекций перед учебными пособиями и традиционным видом лекций состоит в том, что студент может регулировать ее ход самостоятельно, а также возвращаться на предыдущие разделы либо сложные моменты объяснения материала.

Как правило, на экране транслируют видео лектора, но возможно замещением лектора его аватаром (электронным двойником), озвучивающим письменный материал лектора. Максимально эффективными из-за скорости запоминания признаны видеолекции с динамичным изображением, где отображается реальный показ (кинофрагмент, анимация, таблицы) с текстом

Конечно, система онлайн обучения также не идеальна. Во-первых, возникают сложности в определении личности ученика. То есть, пока невозможно точно проверить, сдаёт ли экзамен наш обучающийся либо кто-то за него. Но в качестве решения этой проблемы некоторые вузы дистанционного образования вводят и обязательную очную сессию. Во-вторых, бывает, что качества Интернета недостаточно, чтобы наладить бесперебойную связь между обучающимся и преподавателем. В-третьих, курсы дистанционного обучения способствуют тому, что непосредственный контакт между обучающимся и преподавателем теряется.

Онлайн обучение прекрасно подходит для тех, кто живёт в отдалённых районах, а также для тех, кто в силу определённых причин не может посещать очную форму обучения. Кроме того, несомненным преимуществом дистанционных курсов обучения через Интернет является то, что обучающийся может сам выбрать, в какое время суток ему удобнее заниматься, а также определить для себя индивидуальную продолжительность занятий. Обучающийся дистанционной формы обучения не имеет жёсткого расписания занятий, а все нюансы всегда могут решиться наиболее быстрым образом при помощи электронной почты, скайпа или ICQ. Кроме того, появляется возможность поговорить с преподавателем on-line и задать все интересующие вопросы по тому или иному предмету.

Удаленное образование стоит выбирать целеустремленным людям, имеющим высокую степень мотивации или же тем, кто по каким-то причинам не может получить очное образование. Особенно эффективно такое обучение, если вы уже обладаете азами профессии или специализации и просто хотите [повысить свою квалификацию](http://edunews.ru/kursy/info/kursy-povyshenija-kvalifikacii.html).

Таким образом, анализ научной литературы касаемой изучаемого нами понятия позволяет сделать автору вывод, что не существует единого подхода к определению понятия «онлайн обучение». С вышеизложенного материала видно, что понятия «онлайн обучение» и «электронное обучение» чаще всего рассматриваются как синонимы. Под онлайн обучением автор рассматривает процесс обучения, который доступен или функционирует посредством компьютера или любого другого устройства в котором есть функциональная возможность доступа к сети Интернет. Анализ существующих видов онлайн обучения показывает их разнообразность.

**1.2 Обзор существующих аналогов веб-приложений для онлайн обучения**

**Сэкономить на плате за образование, прослушать лекции знаменитых профессоров, получить повышение квалификации в ведущих университетах мира, воспользоваться обширными электронными библиотеками или поучаствовать в вебинаре – все это становится доступным благодаря онлайн обучению. Мы рассмотрели возможности самых известных мировых образовательных онлайн площадок.**

Наиболее популярными в мире аналогами веб-приложений для онлайн обучения в сети Интернет являются: Courserа, Udacity, EdX, Khan Academy, My Education Path, Canvas Network, Stack Overflow, Uniweb, Универсариум, Университет без границ, HTML Academy, Сетевая Академия Cisco. Автором работы они рассмотрены более подробно.

Coursera – проект в сфере [массового онлайн-образования](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%BE%D0%B2%D1%8B%D0%B9_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D0%BE%D0%BD%D0%BB%D0%B0%D0%B9%D0%BD-%D0%BA%D1%83%D1%80%D1%81), основанный профессорами информатики [Стэнфордского университета](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%8D%D0%BD%D1%84%D0%BE%D1%80%D0%B4%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82) [Эндрю Ыном](https://ru.wikipedia.org/wiki/%D0%AB%D0%BD,_%D0%AD%D0%BD%D0%B4%D1%80%D1%8E) и [Дафной Коллер](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%BB%D0%B5%D1%80,_%D0%94%D0%B0%D1%84%D0%BD%D0%B0). В его рамках существует проект по публикации образовательных материалов в интернете, в виде набора бесплатных онлайн-курсов [7].

Проект сотрудничает с университетами, которые публикуют и ведут в системе курсы по различным отраслям знаний. Слушатели проходят курсы, общаются с сокурсниками, сдают тесты и экзамены непосредственно на сайте Coursera, также распространяется официальное мобильное приложение для [iPhone](https://ru.wikipedia.org/wiki/IPhone) и [Android](https://ru.wikipedia.org/wiki/Android). На ноябрь 2014 года в Coursera зарегистрировано 10 млн пользователей и 844 курса от 108 образовательных учреждений.

В проекте представлены курсы по физике, инженерным дисциплинам, гуманитарным наукам и искусству, медицине, биологии, математике, информатике, экономике и бизнесу. Продолжительность курсов примерно от шести до десяти недель, с 1-2 часами видеолекций в неделю, курсы содержат задания, еженедельные упражнения и иногда заключительный проект или экзамен.

В отличие от таких проектов, как [Academic Earth](https://ru.wikipedia.org/wiki/Academic_Earth), в проекте предлагаются не отдельные лекции, а полноценные курсы, которые включают видеолекции с субтитрами, текстовые конспекты, домашние задания, тесты и итоговые экзамены. Доступ к курсам ограничен по времени; каждое домашнее задание или тест должно быть выполнено только в определенный период времени. По окончании курса, при условии успешной сдачи промежуточных заданий и заключительного экзамена, слушателю выдаётся сертификат об окончании.

По состоянию на 2014 год основная часть курсов представлена на англиском языке, есть курсы на китайском, испанском, францеском, русском, [португальском](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D1%82%D1%83%D0%B3%D0%B0%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) (более десятка), имеются по несколько курсов и на других языках. При этом активно добавляются субтитры на многих языках мира, которые создаются слушателями на добровольных началах. Для создания субтитров на русском языке запущен проект "Переведем Coursera", в котором на начало 2015 года зарегистрировано 15 тысяч участников и переведено 30 курсов [7].

Среди курсов известных лекторов, опубликованных в проекте — «[Машинное обучение](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)» (Эндрю Ын), «Вероятностные графические модели» (Дафна Колер), «[Теория автоматов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%BE%D0%B2)» ([Джеффри Ульман](https://ru.wikipedia.org/wiki/%D0%A3%D0%BB%D1%8C%D0%BC%D0%B0%D0%BD,_%D0%94%D0%B6%D0%B5%D1%84%D1%84%D1%80%D0%B8)), «Принципы [функционального программирования](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) на языке [Scala](https://ru.wikipedia.org/wiki/Scala_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))» ([Мартин Одерски](https://ru.wikipedia.org/wiki/%D0%9E%D0%B4%D0%B5%D1%80%D1%81%D0%BA%D0%B8,_%D0%9C%D0%B0%D1%80%D1%82%D0%B8%D0%BD)), «Руководство для начинающих по иррациональному поведению» ([Дэн Ариэли](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%B8%D1%8D%D0%BB%D0%B8,_%D0%94%D1%8D%D0%BD)), [«Джазовая импровизация»](https://ru.wikipedia.org/w/index.php?title=%D0%94%D0%B6%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D1%8F_%D0%B8%D0%BC%D0%BF%D1%80%D0%BE%D0%B2%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F&action=edit&redlink=1).

В 2012 году Coursera начала работать со Стэндфордом, Принстоном, [Мичиганским](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D1%87%D0%B8%D0%B3%D0%B0%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82) и [Пенсильванским](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D0%BD%D1%81%D0%B8%D0%BB%D1%8C%D0%B2%D0%B0%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82) университетами. 12 образовательных учреждений-партнёров были добавлены в июле 2012 года, и ещё 17 — в сентябре 2012 года. В феврале 2013 года проект сообщил о ещё 29 партнёрских вузах. По состоянию на 2014 год число партнёров — 108. Среди университетов, сотрудничающих с проектом — [Университет Джонса Хопкинса](https://ru.wikipedia.org/wiki/%D0%A3%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82_%D0%94%D0%B6%D0%BE%D0%BD%D1%81%D0%B0_%D0%A5%D0%BE%D0%BF%D0%BA%D0%B8%D0%BD%D1%81%D0%B0), [Калифорнийский технологический институт](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BB%D0%B8%D1%84%D0%BE%D1%80%D0%BD%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82), [Эдинбургский университет](https://ru.wikipedia.org/wiki/%D0%AD%D0%B4%D0%B8%D0%BD%D0%B1%D1%83%D1%80%D0%B3%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82), [Университет Торонто](https://ru.wikipedia.org/wiki/%D0%A3%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82_%D0%A2%D0%BE%D1%80%D0%BE%D0%BD%D1%82%D0%BE), [Колумбийский университет, Пенсильванский университет](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D1%83%D0%BC%D0%B1%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82),  [Московский физико-технический институт](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D1%81%D0%BA%D0%BE%D0%B2%D1%81%D0%BA%D0%B8%D0%B9_%D1%84%D0%B8%D0%B7%D0%B8%D0%BA%D0%BE-%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82), [Высшая школа экономики](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D1%88%D0%B0%D1%8F_%D1%88%D0%BA%D0%BE%D0%BB%D0%B0_%D1%8D%D0%BA%D0%BE%D0%BD%D0%BE%D0%BC%D0%B8%D0%BA%D0%B8), [Новосибирский государственный университет](https://ru.wikipedia.org/wiki/%D0%9D%D0%BE%D0%B2%D0%BE%D1%81%D0%B8%D0%B1%D0%B8%D1%80%D1%81%D0%BA%D0%B8%D0%B9_%D0%B3%D0%BE%D1%81%D1%83%D0%B4%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D1%83%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82).

В конце октября 2013 года Coursera объединилась с [Госдепартаментом США](https://ru.wikipedia.org/wiki/%D0%93%D0%BE%D1%81%D1%83%D0%B4%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B4%D0%B5%D0%BF%D0%B0%D1%80%D1%82%D0%B0%D0%BC%D0%B5%D0%BD%D1%82_%D0%A1%D0%A8%D0%90) для создания обучающих хабов по всему миру. В январе 2014 года Госдепартамент приказал заблокировать доступ к курсам из [Ирана](https://ru.wikipedia.org/wiki/%D0%98%D1%80%D0%B0%D0%BD), [Судана](https://ru.wikipedia.org/wiki/%D0%A1%D1%83%D0%B4%D0%B0%D0%BD) и [Кубы](https://ru.wikipedia.org/wiki/%D0%9A%D1%83%D0%B1%D0%B0).

Около 38,5 % слушателей проживают в США. Такие страны, как [Бразилия](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D0%B7%D0%B8%D0%BB%D0%B8%D1%8F), [Китай](https://ru.wikipedia.org/wiki/%D0%9A%D0%B8%D1%82%D0%B0%D0%B9) и [Индия](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B8%D1%8F), лидируют в списке иностранных слушателей, также на сайте значительная аудитория из [России](https://ru.wikipedia.org/wiki/%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D1%8F), [Германии](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D1%80%D0%BC%D0%B0%D0%BD%D0%B8%D1%8F),  [Испании](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F), [Великобритании](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%BB%D0%B8%D0%BA%D0%BE%D0%B1%D1%80%D0%B8%D1%82%D0%B0%D0%BD%D0%B8%D1%8F), [Канады](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BD%D0%B0%D0%B4%D0%B0), [Австралии](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%81%D1%82%D1%80%D0%B0%D0%BB%D0%B8%D1%8F), [Колумбии](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D1%83%D0%BC%D0%B1%D0%B8%D1%8F), [Украины](https://ru.wikipedia.org/wiki/%D0%A3%D0%BA%D1%80%D0%B0%D0%B8%D0%BD%D0%B0) и [Таиланда](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%B8%D0%BB%D0%B0%D0%BD%D0%B4) [7].

Для курсов доступны [веб-форумы](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%84%D0%BE%D1%80%D1%83%D0%BC), а некоторые слушатели также договариваются о личных встречах для обучения используя [социальные сети](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C), существуют сообщества с регулярными собраниями слушателей. «Кодекс чести» Coursera запрещает копировать ответы, поэтому обсуждения не должны содержать обмена ответами, а только полезные обсуждения.

Udacity – бесплатный сервис, в настоящее включает 14 классов, где «Вы обучаетесь, решая сложные проблемы… со всемирно известными преподавателями университетов». Курсы охватывают темы, которые не только учат вас писать код, но и дают хорошие познания в математике, физике и даже в том «Как построить стартап».

Платформа Udacity – частная разработка, ориентированная, скорее, на изучение материалов IT-тематики ([информатика, программирование](http://ru.osvita.ua/courses/comp/search-46-199-0-898-0.html)). Создала ее группа ученых во главе с Себастьяном Труном, инженером Google, занимающимся робототехникой.

Количество направлений и курсов этой площадки невелико. В отдельные разделы выделены материалы по математике, бизнесу, дизайну, другим наукам (физика, биология). Всего Udacity предлагает около трех десятков курсов [8].

Курсы делятся по уровню сложности: новичок, опытный, профи. На первом уровне предлагается введение в какую-либо науку. Последующие же имеют скорее прикладной характер. Школьники с помощью Udacity могут заработать кредиты для колледжа, студенты – расширить знания об интересующей проблематике (кредиты доступны), профессионалы – приблизить имеющиеся навыки к современным тенденциям.

Видеоряд состоит из слайдов, схем, формул (лектор остается за кадром). Обучение ведется на английском, к некоторым курсам есть субтитры на других языках. Примечательно, что смотреть лекции можно в любое время («дедлайны» при прохождении обучения отсутствуют). По ходу обучения студентам предлагаются викторины и тесты, позволяющие проверить полученные знания. Об успешном окончании курсов свидетельствует сертификат. PDF-файл высылается на почту бесплатно, но можно получить «живую» бумагу, сдав экзамены в одном из центров сертификации. Некоторые работодатели такой диплом рассматривают.

С помощью Udacity можно получить работу. Информация об учениках (по согласию) передается работодателям, заключившим договор с компанией. На данный момент платформа сотрудничает с корпорацией «Google», финансовым гигантом «Bank of America», социальной сетью Facebook и другими компаниями.

Edx.org – некоммерческий проект, основанный Гарвардским университетом и Массачусетским технологическим институтом, предлагающий интерактивное обучение через Интернет. Видеоуроки построены на основе лекций Кембриджского и Массачусетского университетов. По словам руководителей университетов, интернет-платформа будет использоваться не только для создания глобального сообщества онлайн учащихся, но и для поиска методов обучения и технологий. Онлайн курсы дают исследователям возможность отслеживать прогресс студентов, определяя проблемы в системе образования. Здесь можно прослушать кембриджские лекции по искусственному разуму, электронике, программированию, информатике и химии/

EdX – совместный проект Гарвардского университета и МТИ, предлагающий пройти онлайн-обучение по следующим направлениям: компьютерные технологии, химия, математика, этика, музыка, статистика, литература (всего 24 направления). Онлайн-курсы повторяют реальные лекции, которые читаются в Гарварде, университете Корнуэлла, Федеральной политехнической школе Лозанны, Гонконгском, Киотском, Пекинском университетах, поэтому обучающиеся должны иметь соответствующие базовые знания. Некоторые курсы засчитываются в академические часы, могут учитываться при рассмотрении заявки на получение стипендии в университетах-партнерах [9].

Курсы на edX делятся на модули. По ходу их изучения студентам даются домашние задания (результаты выполнения работ засчитываются в общий балл). Для контроля знаний проводятся экзамены (один курс может содержать несколько экзаменов, каждый из которых также влияет на конечную оценку). Успешные студенты по окончании курса получают сертификат.

В помощь обучающимся к каждому курсу прилагаются онлайн-учебники, дается возможность обсуждать материалы и задания на форуме. Скорость видеолекций можно регулировать (при неудовлетворительном уровне английского можно замедлить темп речи и понять, что говорит лектор).

Прогресс и полученные баллы можно отслеживать в специальном разделе аккаунта.

[Khan Academy](http://www.khanacademy.org/) – огромная база видео-уроков на более чем 20 языках, все курсы разделены на отдельные уроки с возможностью просмотра каждого видео независимо от того, какой курс вы выбрали. Основной акцент сделан на видео как способ предоставления информации, материалов для чтения здесь нет.

Проект My Education Path не предлагает курсы самостоятельно, лишь собирает и структурирует информацию о лекциях, читаемых на других онлайн-площадках. Помимо этого, доступны и наборы курсов по разным направлениям, которые вместе составляют целостную программу. Есть желание изучать Computer Science? Пожалуйста. Психологию? Нет проблем. Решили освоить Высшую математику? Есть варианты.

Впрочем, каждый семестр предлагается солидная нагрузка, так что будьте готовы к тому, что вам понадобится до 30–40 часов в неделю для успешного освоения материала.

Проект Canvas Network отличается большим разнообразием курсов, которые проводят абсолютно разные по уровню подготовки и направлению деятельности люди: доктора наук, менеджеры, писатели. Курсы не имеют единого подхода к изложению. Материал могут объяснять исключительно в коротких видеолекциях, дополнять возможностью обсуждать прослушанное на форуме с преподавателем и другими студентами, сертификат по окончании курса могут выдавать или нет. Особенности конкретного курса содержатся в описании. Кроме вышеизложенной информации в превью можно узнать, для кого разработана данная программа (доступна для всех, ориентирована исключительно на профессионалов в той или иной отрасли). Курсы могут иметь возрастные ограничения.

Canvas Network предлагает бесплатные, условно бесплатные и платные курсы. Условно бесплатные предполагают покупку дополнительных учебных материалов (пособий, литературы), платные позволяют заработать кредиты в системе непрерывного образования (актуально для профессий, где необходимо постоянно подтверждать свою квалификацию – учителей, архитекторов, медицинских работников и т. д.) [9].

Курсы длятся 2–3 недели, анонсируются за месяц и ранее, что позволяет желающим предварительно записаться на них. Поскольку количество мест на курсе ограничено, в интересных классах лучше регистрироваться как можно раньше.

[Stack Overflow](http://stackoverflow.com/) – хоть он технически и не содержит «туториалов», но имеет тонны (легко искомой) информации, которая может здорово помочь. Также, если вы на чем-то застряли (и на ресурсе еще нету ответа), сообщество очень хорошо отвечает на вопросы.

Подавляющее число курсов ведется на английском языке, но в последнее время стали появляться программы интерактивного обучения на русском языке. Приведем анализ нескольких крупнейших платформ, которые дают возможность послушать лекцию, поучаствовать в вебинаре или пройти целый обучающий курс на русском языке.

Uniweb – это платформа, где собраны онлайн-курсы нескольких крупных университетов, среди которых МГИМО, РАНХиГС и МГУ. Здесь, в основном, представлены программы по бизнес-образованию продолжительностью от одной до восьми недель. За это время учащиеся должны просмотреть курс небольших видеолекций, изучить сопутствующие материалы и литературу, а также выполнить домашние задания. Слушатели могут сами планировать свое расписание, однако, у курса есть дедлайн и на выполнение заданий дается строго определенное количество времени [10].

Стоимость программ может быть от трех до двадцати тысяч рублей, в зависимости от количества учебных часов. Но, например, трехнедельный курс «Современная PR-служба», подготовленный МГИМО, можно пройти бесплатно. К бесплатным занятиям можно приступать в любой момент, а чтобы поступить на платный курс надо будет отправить по электронной почте пакет документов для зачисления. По окончании занятий слушатели получают сертификаты.

«Универсариум» открылся совсем недавно: первые занятия в нем начались только в январе этого года. В будущем он должен стать местом для взаимодействия работодателей и специалистов, поэтому уже сейчас часть курсов ориентирована на конкретных предпринимателей и дает возможность трудоустройства для самых успешных выпускников.

Обучение в «Универсариуме» бесплатное и может продолжаться от семи до десяти недель. Программы, разработанные преподавателями МГУ и МФТИ, состоят из модулей, а модули, в свою очередь, включают в себя видеолекцию, самостоятельную работу, домашнее задание и тест. Сейчас на сайте доступны программы по химии, математике и микроэкономике. В ближайшее время появятся курсы для гуманитариев.

Одна из главных особенностей «Универсариума» — система кросс-проверки домашних работ. Каждое выполненное задание просматривают не только преподаватели, но и другие учащиеся, а итоговая оценка рассчитывается как средний балл. За проверку чужих заданий также можно получить дополнительные баллы [10].

В ближайшей перспективе выпускники «Универсариума» смогут получать дипломы. А пока оценка знаний отображается в общем рейтинге студентов.

Главное отличие «Университета без границ» от других систем онлайн-обучения – это то, что курсы проходят в режиме реального времени, поэтому учащиеся могут не только послушать лекцию, но и задать вопросы, спросить совет или устроить дискуссию.

Разработчики учебных программ делали основной упор на медиа, историю, политику, глобализацию, урбанистику, социологию. Курсы ведут преподаватели американских, европейских и российских вузов, включая Tufts University, Georgetown University, George Mason University, Мах Planck Institute, МВШСЭН.

Стоимость большинства программ – 100$, но есть также и бесплатные курсы. По окончании обучения студентам, которые выполнили не менее 70% заданий, выдается электронный сертификат [10].

Интерактивные курсы HTML Academy посвящены обучению HTML и CSS. Основной упор делается на практических занятия, поэтому программа состоит из многочисленных упражнений, в ходе которых ученику необходимо изменять HTML и CSS код в онлайн редакторах, следуя указаниям системы. Учебная программа состоит из восьми тем, изучению которых посвящен один или несколько курсов. В конце ученикам необходимо выполнить итоговое испытание – сверстать макет, специально разработанный с учетом пройденного материала.

Сейчас большинство курсов на сайте бесплатные, но за 100 российских рублей в месяц можно оформить подписку, которая дает моментальный доступ ко всем новым материалам, а также к продвинутым курсам по HTML5 и CSS3.

Программа Сетевой Академии Cisco – это комплексная программа электронного обучения, предоставляющая студентам знания в области технологий Интернета, необходимые в условиях глобальной экономики. Программа Сетевой Академии включает материалы, доступные через Интернет, инструменты оценки знаний, средства отслеживания академических успехов студентов, практические лабораторные занятия, а также курсы подготовки для получения признанных в отрасли профессиональных сертификатов.

Сетевая Академия была основана в октябре 1997 г. на базе 64 учебных заведений в семи штатах и к настоящему времени распространилась на более чем 150 стран. С момента основания программы более 1.6 млн. студентов поступили в более 10 000 Академий, расположенных в школах, техникумах, колледжах, университетах и общественных организациях.   
Заинтересованным учебным заведениям присваивается статус Сетевой Академии уровня, соответствующего уровню обучения, которое они будут предоставлять в рамках программы. В настоящее время существует три возможных уровня обучения. Эксперты Cisco Systems обучают Тренеров преподавателей в Центрах подготовки Академии Cisco (CATC). Тренеры, получившие подготовку в CATC, обучают преподавателей Региональных Академий (Regional Academies), а они, в свою очередь, обучают преподавателей Местных Академий (Local Academies), которые и преподают студентам. Использование этой трехуровневой модели обучения позволяет обеспечить необходимый уровень подготовки преподавателей в непосредственной близости от мест их проживания. Учебные заведения могут принимать участие в обучении на одном или нескольких уровнях.   
Партнеры Cisco из числа коммерческих, правительственных и общественных организаций формируют экосистему, предоставляющую ряд услуг и поддержку, необходимую для подготовки будущих сотрудников в условиях глобальной экономики. Учебная программа, изначально созданная для подготовки для получения сертификатов "Сертифицированный Cisco Сетевой Специалист" (CCNA) и "Сертифицированный Cisco Сетевой Профессионал" (CCNP), значительно расширилась за счет учебных курсов, поддерживаемых организациями-партнерами. Дополнительные курсы включают: "Основы ИТ: Оборудование и программное обеспечение ПК" (IT Essentials: PC Hardware and Software) и "Основы ИТ: Сетевые операционные системы" (IT Essentials: Network Operating Systems); а также курс "Основы сетевой инфраструктуры Panduit" (Panduit Network Infrastructure Essentials) при поддержке Panduit Corporation.

Использование Интернета позволяет предоставлять равный уровень обучения для всех студентов независимо от их местоположения, социально-экономического статуса, пола или расы. При поддержке Программы развития Организации объединенных наций, Агентства США по международному развитию и международного союза по телекоммуникациям Cisco обеспечила доступ к программе студентов из наименее развитых стран, чтобы помочь им строить экономики своих стран. Программа Сетевой Академии постоянно повышает стандарты учебного процесса и дистанционного обучения. Благодаря обратной связи и электронным средствам оценки, программа Академии постоянно адаптируется с целью повышения успешности образования и достижений студентов. Инфраструктура Академии позволяет предоставлять богатую, интерактивную и персонализированную программу обучения студентам во всем мире. Интернет способен изменить способы обучения, работы и развлечения для всех нас, и программа Сетевой Академии Cisco находится на переднем крае этой трансформации [11].

Таким образом, обзор существующих аналогов веб-приложений для онлайн обучения показывает их многообразие и многофункциональность. На сегодняшний день, данная область развивается очень стремительно. С каждым годом появляются новые веб-приролежения, более усовершенствованные, что позволяет автору утверждать о популярности использования онлайн обучения. Несмотря на это данная область в Республике Беларусь требует большего развития и новых разработо.

[**2 РАЗРАБОТКА ТРЕБОВАНИЙ И ФУНКЦИОНАЛЬНОЙ МОДЕЛИ**](#_Toc452938679)

**2**[**.1 Выбор языка программирования, платформ и средств разработки ПС**](#_Toc452938677)

Приложение реализовано на языке программирования Ruby с использованием фреймворка Rails, языка разметки гипертекста HTML, каскадных таблиц стилей CSS и JavaScript-фреймворка Angularjs. В качестве среды разработки была использована IDE RubyMine 8. Для создания и работы с базой данных была использована СУБД PostgreSQL. Рассмотрим преимущества.

Ruby – динамический, рефлективный, интерпретируемый высокоуровневый язык программирования для быстрого и удобного объектно-ориентированного программирования. Язык обладает независимой от операционной системы реализацией многопоточности, строгой динамической типизацией, сборщиком мусора и многими другими возможностями. По особенностям синтаксиса он близок к языкам Perl и Eiffel, по объектно-ориентированному подходу – к Smalltalk. Также некоторые черты языка взяты из Python, Lisp, Dylan и Клу [7].

Кроссплатформенная реализация интерпретатора языка является полностью свободно. Ruby – полностью объектно-ориентированный язык. В нём все данные являются объектами, в отличие от многих других языков, где существуют примитивные типы. Каждая функция – метод.

Ruby использует вызов по соиспользованию (call-by-sharing), хотя в сообществе Ruby часто говорят, что он использует вызов по ссылке. Для программиста, привыкшего к распространённым гибридным языкам программирования, некоторые эффекты такого решения могут показаться неожиданными. Ruby не поддерживает множественное наследование, но вместо него есть мощный механизм примесей. Все классы (напрямую или через другие классы) выведены из класса «Object», следовательно, любой объект может использовать определённые в нём методы. Процедурный стиль также поддерживается, но все глобальные процедуры неявно являются закрытыми методами класса «Object».

Ruby является мультипарадигменным языком: он поддерживает процедурный стиль (определение функций и переменных вне классов), объектно-ориентированный, функциональный (анонимные функции, замыкания, возврат значения всеми инструкциями, возврат функцией последнего вычисленного значения). Он поддерживает отражение, метапрограммирование, информацию о типах переменных на стадии.

Ruby on Rails – полноценный, многоуровневый фреймворк для построения веб-приложений, использующих базы данных, который основан на архитектуре Модель-Представление-Контроллер (Model-View-Controller, MVC).

Является открытым программным обеспечением и распространяется под лицензией MIT.

Базируется на следующих принципах разработки приложений:

* максимальное использование механизмов повторного использования, позволяющих минимизировать дублирование кода в приложениях (принцип Don’t repeat yourself);
* по умолчанию используются соглашения по конфигурации, типичные для большинства приложений (принцип Convention over configuration) — явная спецификация конфигурации требуется только в нестандартных случаях [12].

RubyMine – умная и удобная интегрированная среда разработки для языка Ruby и веб-фреймворка Rails, которая включает в себя все необходимые разработчику инструменты и поддерживает наиболее востребованные технологии из мира Ruby.

Как и все IDE, разработанные на основе платформы IntelliJ, RubyMine обладает уникальным механизмом анализа кода, который лежит в основе таких функций, как интеллектуальное автодополнение кода, рефакторинг, исправление кода на лету и навигация по коду. Особенностью RubyMine является то, что для всех перечисленных выше функций учитывается специфика каждого конкретного проекта.

Так, к примеру, для Rails-приложения переименование модели приведет к переименованию соответствующих контроллера и файлов представлений, а также к созданию миграционного файла для внесения необходимых изменений в базу данных. Навигация позволит быстро перейти от метода контроллера к файлу представления и обратно, или к соответствующим тестовым сценариям. В случае RubyMotion-приложения, автодополнение кода работает не только для языка Ruby, но и для кода на Objective-C, причем не только в редакторе, но и в консоли [12].

Кроме того, RubyMine позволяет полностью настроить рабочую среду, отвечающую требованиям разработчика, начиная с цветовой схемы, расположения окон, сочетания клавиш быстрого доступа и заканчивая интеграцией с системой контроля версий, базой данных, или настройкой удаленного сервера.

Таким образом, рассмотрим основные приемущества Ruby / Ruby on Rails:

Скорость разработки. Основным преимуществом языка программирования Ruby и фреймворка Ruby on Rails считается скорость разработки. Практика показывает, что скорость разработки проектов на RoR увеличивается на 30–40 процентов по отношению к любому другому языку программирования или фреймворку. В первую очередь прирост скорости разработки определяется обширным набором готовых к работе штатных инструментов RoR, колоссальным набором готовых решений в сообществе, языку Ruby и простоте программирования на нем. Одной из важнейших частей культуры RoR является социальность. Решил проблему, — помоги решить другим. Реализовал свой модуль, — поделись с сообществом. Таким образом, на данный момент в открытом доступе хранятся тысячи готовых решений тех или иных задач. Системы аутентификации, авторизации, комментирования, системы платежей, почтовые рассылки и многое другое (все то, что обычно разрабатывается «с нуля») внедряются реализованные кем–то другим, протестированные и рекомендованные многочисленным сообществом.

Культура и стандарты. Ruby on Rails — это фреймворк. Зачастую фреймворк не позволяет вам самодеятельность. Конечно же, в Ruby on Rails можно «изобрести свой велосипед» и программировать в любых направлениях, не опираясь на стандарты; но зачастую этого не требуется. Стандарты размещения файлов в проекте, стандарты написания кода в проекте, общие правила программирования в Ruby on Rails сильно структурируют любой проект. За счет этого проект становится читаемым. Вхождение в проект новичков происходит очень быстро. Опыт показывает, что любой новичок в проекте в первый же день работы делает свои первые полезные правки. За счет этого не считается большой проблемой, если разработку проекта изначально вела одна команда программистов, а поддержку проекта или доработку — совершенно другая. Проект на RoR априори понятен любому разработчику.

Тестирование. При разработке любого крупного проекта встает резонный вопрос. Как и кто будет тестировать проект? Не всегда есть средства и желание создавать целые отделы тестирования, к тому же хочется автоматизировать этот процесс. В отличие от других фреймворков, в составе RoR есть отличные средства автоматизированного тестирования. В других языках программирования и фреймворках штатных средств тестирования нет. Конечно, есть сторонние разработки, позволяющие организовать автоматическое тестирование проекта на PHP, но они не ставятся “из коробки” и об их использовании программисты чаще не задумываются. В проекте на Ruby on Rails, в идеале, код проекта не пишется до тех пор, пока под этот код не написаны тесты. RoR идеология предполагает изначальное использование методов BDD (Behavior Driven Development) или TDD (Test Driven Development).

Кеширование. Кеширование проектов — один из важнейших этапов разработки крупного интернет–проекта. В PHP имеются различные варианты кеширования данных. Эти варианты и инструменты прикручиваются, пристраиваются, приделываются, прикрепляются сбоку. До сих пор в сообществе PHP нет единого мнения: что лучше использовать, как лучше кешировать данные, какими инструментами пользоваться. Ruby on Rails в его базовой комплектации имеет штатные средства кеширования данных. На старте предоставляются инструменты, позволяющие реализовать кеширование данных на проекте. Вы можете кешировать целые страницы или же блоки кода. Можете кешировать результаты запросов и ActiveRecord–модели. Кешировать можно как при помощи memcached или redis, так и другими средствами. Для реализации кеширования на Ruby on Rails проекте вам в 95 процентах случаев не потребуется ничего кроме уже готовых и штатных решений.

Локализация. Часто встречается ситуация, когда кто-то сделал проект, а потом неожиданно понимает, что для продолжения развития проекта необходима английская версия. Ruby on Rails в базовой комплектации имеет средства локализации проекта. Вы можете предусмотреть необходимость поддержки различных языков на сайте как изначально, так и в дальнейшем. RoR умеет раздавать разные шаблоны для разных языков, содержит в себе конфигурационные файлы с переводами терминов и многие другие штатные инструменты для реализации локализации проекта.

Роутинг (красивые урлы или ЧПУ). Зачастую во многих PHP проектах мы можем видеть картину, когда адрес определенной страницы огромен и непонятен. В Ruby on Rails есть штатная возможность гибко настроить ваш роутинг, вид адресов, названия основных разделов. Есть возможность быстро изменить адреса в одном месте без необходимости изменения этого адреса во всем проекте. В сообществе RoR–разработчиков активно используются идеология REST. Адреса страниц в проектах на Ruby on Rails всегда понятны, красивы, прекрасно понимаются поисковиками, просты.

Валидации. В ruby on rails прекрасно реализованы инструменты, позволяющие валидировать входящие данные. Ваши пользователи заполняют формы и нужно проверить правильность ввода адреса электронной почты, наличие пароля или необходимую минимальную длину логина, — штатные средства Rails вам в этом помогут.

Миграции и работа с базой данных. Обыденная проблема многих проектов на PHP — невозможность понятными средствами и инструментами контроллировать структуру базы данных. Изменения в структуру зачастую вносятся вручную и прямо в базу. Из–за этого в проекте появляются многочисленные непонятные поля и таблицы, про которые уже никто ничего не помнит. В Ruby on Rails существуют штатные инструменты работы с базами данных — «миграции». Структура базы данных хранится в коде приложения и конфигурируется из проекта. Ваша структура будет всегда в репозитории, любое изменение структуры будет задокументировано и привязано к определенному коммиту в репозиторий.

Безопасность. Ruby on Rails по умолчанию сильно заточены под безопасность проекта. При использовании инструментов RoR исключены SQL инъекции и XSS атаки. Все входные параметры экранируется по умолчанию. Выводимые переменные в шаблонах также экранируются, только если вы не указали обратной опции. У разработчика нет шансов допустить ошибки безопасности (не без исключений, разумеется).

Деплой. В среде Ruby on Rails существует много удобных и приятных инструментов. В том числе инструменты, которые применяются в процессе деплоя. Например, используя Capistrano, выкатка новой версии приложения на боевой сервер (или несколько серверов) потребует одной команды в консоли: cap deploy [12].

Автором работы также рассмотрены дополнительные принципы разработки на Ruby / Ruby On Rails, такие как: системы контроля версий, и сисстемы управления проектами.

При разработке любого Ruby on Rails проекта подразумевается использование известных систем контроля версий. Использование git, как говорится, «добровольно–принудительно», так как многие системы автоматического развертывания проекта на «боевых» серверах не работают без них. Программисты на RoR изначально, при изучении платформы, вынуждены осваивать git, так как многочисленные примеры кода в документации подразумевают использования данных систем контроля версий. Во многом из за этого неопытным новичкам проще начать изучать PHP и не трогать Rails до достижения определенного уровня понимания веб–разработки как таковой и ее принципов.

Ruby on Rails был изначально разработан для того, чтобы реализовать систему управления проектом — Basecamp. Также на RoR был создан Redmine (популярная и бесплатная система управления проектом). Поэтому при работе над Rails проектами «добровольно–принудительно» использование таких систем. Все системы интегрируются с системами контроля версий, что позволяет более гибко регулировать процессы разработки проекта.

Таким образом, выбор языка програмирования, платформ и средств разработки ПС, базируется на функциональных возможностях и приемуществах программного продукта. Основными приемуществами Ruby / Ruby on Rails являются: скорость разработки, тестирование, кеширование, локализация, роутинг, миграции и работа с базой данных, безопасность, деплой. Поэтому выбор языка программирования вполне обоснован.

**2.2 Разработка функциональной модели**

Для представления функциональной модели была выбраны диаграмма вариантов использования UML и диаграмма деятельности UML.

Диаграмма вариантов использования описывает функциональное назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования. Диаграмма вариантов использования является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

**Диаграмма деятельности** – это технология, позволяющая описывать логику процедур, бизнес-процессы и потоки работ. **Диаграмма деятельности** позволяет любому, кто выполняет данный процесс, выбирать порядок действий. Другими словами, диаграмма только устанавливает правила обязательной последовательности действий, которым необходимо следовать. В данном веб-приложении диаграмма деятельности представляет собой возможные действия пользователя с ролью «Слушатель» в динамике и логической последовательности, которые в совокупности приводят к получению желаемого результата.

**2.2.1 Действующие лица**

Множество систем имеет разделение на категории пользователей. Поэтому каждая категория пользователей представляется отдельным действующим лицо (актером).

При анализе работы системы были выделены следующие действующие лица и разработаны варианты использования:

Лектор – добавлять и удалять учебные курсы, которые он разместил, добавлять и удалять видео лекции к курсам, которые он разместил, добавлять и удалять дополнительные текстовые материалы, которые он разместил, редактировать информацию о себе.

Незарегистрированный пользователь – может просмотреть главную страницу с курсами, может зарегистрироваться или войти на сайт.

Зарегистрированный пользователь – отправка личных сообщений другим пользователям, редактирование личных данных.

Слушатель – просматривать, оценивать и комментировать видео лекции, скачивать дополнительный материал курсов, подписываться на понравившиеся курсы.

Администратор – ведет управление категориями, пользователями и учебными курсами на сайте.

Организация взаимодействия между пользователями должна осуществляться путем введения модуля сообщений и модуля комментариев. Благодаря этим частям приложения осуществляется общение слушателей и лекторов.

**2.2.2 Варианты использования**

Каждый вариант использования определяет последовательность действий, которые должны быть выполнены проектируемой системой при взаимодействии ее с соответствующим актером. Диаграмма вариантов может дополняться пояснительным текстом, который раскрывает смысл или семантику составляющих ее компонентов. Такой пояснительный текст получил название примечания или сценария.

Для разрабатываемой системы, исходя из потребностей действующих лиц, можно выделить следующие варианты использования:

* аутентификация;
* регистрация;
* добавление учебного курса;
* удаление учебного курса;
* добавление видео лекции;
* удаление видео лекции;
* добавление дополнительных текстовых материалов курса;
* удаление дополнительных текстовых материалов курса;
* редактирование информации о себе;
* подписка на учебный курс;
* комментарий к видео лекции;
* оценка видео лекции («понравилась», «не понравилась»)
* отправка текстового сообщения.

**2.2.3 Диаграмма вариантов использования**

На диаграмме вариантов использования показывают взаимодействия между всеми действующими лицами и вариантами использования. Диаграмма должна показывать, какие действующие лица инициируют варианты использования, а также должна отображать, когда действующие лица получают информацию от вариантов использования.

Основные взаимодействия между действующими лицами и вариантами использования задаются с помощью связи коммуникации. Задается в виде простой стрелки. Направление стрелки показывает, кто инициирует связь (всегда действующее лицо) и какой вариант использования отправляет информацию внешнему действующему лицу.

Каждый вариант использования показывает, как конкретный актер использует систему. Для того чтобы начать выполнять операции, в системе необходимо пройти аутентификацию, с целью установления прав пользователей.

Автором рассмотрены варианты использования.

Для незарегистрированного пользователя:

А. Вариант использования «Аутентификация»:

* назначение: данный вариант использования описывает вход пользователя в систему;
* основной поток событий: данный вариант используется, когда пользователь хочет войти в систему:

1) система запрашивает электронную почту пользователя и пароль;

2) пользователь вводит почту и пароль;

3) система проверяет почту и пароль, после чего открывает доступ в систему.

* альтернативный поток: неправильная почта или пароль. Если во время выполнения основного потока обнаружится, что пользователь ввел неправильную почту или пароль, то система выводит сообщение об ошибке. Пользователь может вернуться к началу основного потока или отказаться от входа в систему (при этом выполнение варианта использования завершается);
* предусловия: отсутствуют;
* постусловия: если вариант использования выполнен успешно, пользователь входит в систему. В противном случае состояние системы не изменится.

Б. Вариант использования «Регистрация»:

* назначение: данный вариант использования описывает регистрацию пользователя в системе;
* основной поток событий: данный вариант используется, когда пользователь хочет зарегистрироваться в системе:

1) система запрашивает электронную почту, пароль, а также подтверждение пароля, имя, фамилию, пол и дату рождения;

2) пользователь вводит электронную почту, пароль, подтверждение пароля, имя, фамилию, пол и дату рождения;

3) система проверяет почту, пароль, имя, фамилию, дату рождения и после чего открывает доступ в систему.

* альтернативный поток: неправильная электронная почта, пароль, подтверждение пароля, имя, фамилия, дата рождения. Если во время выполнения основного потока обнаружится, что пользователь ввел неправильно электронную почту, пароль, подтверждение пароля, имя, фамилию, дату рождения, то система выводит сообщение об ошибке. Пользователь может вернуться к началу основного потока или отказаться от входа в систему (при этом выполнение варианта использования завершается). Если пароль подтвержден неверно, то система выводит сообщение о ошибке;
* предусловия: отсутствуют;
* постусловия: если вариант использования выполнен успешно, пользователь перенаправляется на форму входа в систему. В противном случае состояние системы не изменится.

Для любого зарегистрированного пользователя:

А. Вариант использования «Отправка личных сообщений»:

* назначение: данный вариант использования описывает отправку личных сообщений другим пользователям;
* основной поток событий: данный вариант использования начинает выполняться, когда зарегистрированный пользователь хочет отправить личное сообщение другому пользователю и заходит во вкладку «Сообщения»:

а) пользователь кликает на кнопку «Новое сообщение»;

б) пользователь вводит данные и отправляет сообщение;

в) система проверяет корректность введенной информации;

г) система сохраняет сообщение в базе данных.

Б. Вариант использования «Написание комментария к видеолекции»:

* назначение: данный вариант использования предназначен для написания комментария к видеолекции учебного курса;
* основной поток событий: данный вариант использования начинает выполняться, когда зарегистрированный пользователь выбрал видеолекцию учебного курса и хочет прокомментировать ее:

а) пользователь кликает на кнопку «Новый комментарий»;

б) пользователь вводит комментарий;

в) система проверяет корректность введенной информации;

г) система сохраняет комментарий в базе данных.

В. Вариант использования «Редактирование личных данных»:

* назначение: данный вариант использования служит для редактирования личных данных зарегистрированных пользователей;
* основной поток событий: данный вариант использования начинает выполняться, когда зарегистрированный пользователь хочет редактировать личные данные:

а) выбор вкладки «Мой профайл»;

б) ввод новых данных пользователем;

в) система проверяет корректность введенной информации;

г) система сохраняет новые данные в базу данных.

* альтернативные потоки:

1. не заполнено обязательное поле или заполнено неверно.

Для зарегистрированного пользователя c ролью «Слушатель»:

А. Вариант использования «Просмотр видеолекции учебного курса»:

* назначение: данный вариант использования описывает просмотр пользователем видеолекции учебного курса;
* основной поток событий: данный вариант использования начинает выполняться, когда пользователь с ролью слушатель выбрал учебный курс:

а) пользователь выбирает нужную видеолекцию;

б) с помощью встроеннго видеоплеера пользователь просматривает видеолекцию;

Б. Вариант использования «Оценка видеолекции»:

* назначение: данный вариант использования предназначен для описания процесса оценки видеолекции учебного курса (понравилась видеолекция или нет);
* основной поток событий: данный вариант использования начинает выполняться, когда пользователь с ролью слушатель выбрал видеолекцию учебного курса и хочет ее оценить:

a) пользователь выбирает видеолекцию;

б) пользователь оценивает видеолекцию с помощью клика по кнопкам «понравилось» или «непонравилось»;

в) система увеличивает количество голосов (понравилось либо непонравилось) для данной видеолекции;

В. Вариант использования «Скачивание дополнительного материала»:

* назначение: данный вариант использования служит для скачивания дополнительного материала;
* основной поток событий: данный вариант использования начинает выполняться, когда пользователь с ролью слушатель выбрал учебный курс и хочет получить дополнительный материал курса:

а) выбор нужного дополнительного материала(файла);

б) скачивание файла пользователю;

Г. Вариант использования «Подписка на учебный курс»:

* назначение: данный вариант использования служит для оформления подписки на выбранный учебный курс;
* основной поток событий: данный вариант использования начинает выполняться, когда пользователь с ролью слушатель выбрал учебный курс и хочет подписаться на него:

а) пользователь кликает на кнопку «Подписаться»;

б) система производит подписку пользователя на выбранный учебный курс;

Для зарегистрированного пользователя c ролью «Лектор»:

А. Вариант использования «Действия с учебным курсом»:

* назначение: данный вариант использования служит для добавления, удаления, редктирования учебного курса;
* основной поток событий: данный вариант использования начинает выполняться, когда пользователь входит в систему под ролью «лектор»:

1) добавление учебного курса:

а) пользователь вводит данные для нового учебного курса;

б) система проверяет корректность введенной информации;

в) система добавляет курс в базу данных и отображает его в учебных курсах пользователя.

2) редактирование учебного курса:

а) система предоставляет специальную форму для редактирования своего учебного курса;

б) пользователь вводит данные;

в) система проверяет корректность введенной информации;

г) система обновляет данные об учебном курсе в базе данных.

3) удаление учебного курса:

а) пользователь выбирает нужный учебный курс;

б) пользователь производит удаление курса (включая все его соержимое);

в) система производит удаление записи об учебном курсе и вносит изменения в базу данных.

Б. Вариант использования «Действия с видеолекцией»:

* назначение: добавления, удаления видеолекций в учебный курс;
* основной поток событий: данный вариант использования начинает выполняться, когда пользователь входит в систему под ролью «лектор» и выбирает определенный учебный курс:

1) добавление видеолекции в учебный курс:

а) пользователь выбирает пункт «Добавить видеолекцию»;

б) пользователь вводит данные и загружает файлы;

в) система производит добавление записи в базу данных и отображает видео в учебном курсе.

2) удаление видеолекции из учебного курса:

а) пользователь выбирает нужную видеолекцию;

б) пользователь производит удаление видеолекции;

в) система проверяет корректность информации;

г) система производит удаление записи о видеолекции и вносит изменения в базу данных.

В. Вариант использования «Действия с дополнительным материалом к учебному курсу»:

* назначение: добавления, удаления видеолекций; добавления, удаления учебных материалов(документов);
* основной поток событий: данный вариант использования начинает выполняться, когда пользователь входит в систему под ролью «лектор» и выбирает определенный учебный курс:

1) добавление дополнительного материала в учебный курс:

а) пользователь выбирает пункт «Добавить дополнительный материал»;

б) пользователь вводит название и загружает файлы;

в) система производит добавление записи в базу данных и отображает файлы в учебном курсе.

2) удаление дополнительного материала из учебного курса:

а) пользователь выбирает нужный дополнительный материал;

б) пользователь производит удаление дополнительного материала;

в) система проверяет корректность информации;

г) система производит удаление записи о дополнительном материале и вносит изменения в базу данных.

* альтернативные потоки:

1) не заполнено обязательное поле или заполнено неверно.

Для пользователя c ролью «Администратор»:

А. Вариант использования «Действия администратора»:

* назначение: служит для добавления, удаления и обновления информации, которая необходима для функционирования системы;
* основной поток событий: выполняется, когда «Администратор» начинает работу с системой: «Администратор» выбирает действие, начинает выполняться один из подчиненных потоков: добавление, обновление и удаление данных.

1) добавление данных:

а) «Администратор» выбирает соответствующий раздел;

б) система запрашивает соответствующие данные для ввода;

в) «Администратор» вводит данные;

г) система проверяет корректность введенных данных;

д) система сохраняет данные по соответствующему разделу в базу данных.

2) обновление данных:

а) «Администратор» выбирает соответствующий раздел;

б) система запрашивает соответствующие данные для обновления;

в) «Администратор» указывает соответствующие данные;

г) система выводит соответствующие данные для обновления;

д) «Администратор» обновляет данные;

е) система проверяет корректность данных и сохраняет их в базе данных.

3) удаление данных:

а) «Администратор» выбирает соответствующий раздел;

б) система запрашивает соответствующие данные для удаления;

в) «Администратор» указывает соответствующие данные;

г) система удаляет данные.

* альтернативные потоки: если добавление, удаление, обновление информации по каким-либо причинам невозможно, то выводится соответствующее сообщение, содержащее причину ошибки и возможности ее разрешения:

1) добавление данных: если данные для добавления некорректны, то система выводит предупреждающее сообщение о некорректности данных;

2) обновление данных: если данные для обновления некорректны, то система выводит предупреждающее сообщение о некорректности данных;

3) удаление данных: если удаляемые данные имеют связь с другими данными, то система выводит предупреждающее сообщение о невозможности удаления данных.

* предусловия: перед началом выполнения данного варианта использования администратор должен войти в систему;
* постусловия: если вариант использования завершится успешно, то данные в базе будут обновлены соответствующим образом. Если же в процессе выполнения произошли какие-либо ошибки, то обновления в базе не сохраняются.

**2.3 Общие требования к разрабатываемому веб-приложению**

**2.3.1 Общие требования**

Веб-приложение предназначено для широкого круга пользователей и будет использоваться для размещения обучающих курсов по любым научным направлениям. Взаимодействие с приложением должно осуществляться через глобальную сеть с использованием протоколов прикладного уровня. Программный продукт должен отвечать следующим требованиям:

* доступность: способность предоставлять доступ к учебным материалам из точки удаленного доступа;
* кроссплатформенность: не имеет значения, какой программной средой или аппаратной платформой располагает пользователь, для работы с приложением необходим только браузер;
* совместимость с основными версиями браузеров.
* долговечность: способность соответствовать новым технологиям без дополнительной и дорогостоящей доработки;
* надежность: обеспечение надежности хранения данных при сбое аппаратуры клиентских компьютеров.

**2.3.2 Требования к модулю накопления данных**

В системе необходимо организовать механизмы хранения учебных курсов, способы их описания и поиска. Все добавленные данные должны быть четко структурированы и представляться в виде курса. Курс должен состоять из видеолекций и иметь приложенные дополнительные материалы и задания. Поиск учебных курсов должен осуществляться по имени и по категории курса, сортировка – по дате добавления.

**2.3.3 Требования к пользовательскому интерфейсу**

* функциональность (соответствие задачам пользователя);
* соответствие технологии;
* понятность и логичность;
* обеспечение высокой скорости работы пользователя;
* обеспечение защиты от человеческих ошибок;
* быстрое обучение пользователя;
* субъективное удовлетворение пользователя.

**2.3.4 Требования к возможностям модернизации**

В приложении должна быть реализована возможность модернизации и добавления дополнительной функциональности. Модель приложения должна позволять адаптировать ее к изменяющимся в процессе его эксплуатации условиям и допускать поэтапную модернизацию отдельных компонентов. Для достижения этих целей в процессе программирования необходимо использовать объектно-ориентированный стиль, использовать REST API архитектуру и соблюдать требования к процессу программирования.

**2.3.5 Технические требования к реализации системы**

Физическую реализацию требуется организовать на ЭВМ с операционной системой Linux. Рекомендуемые сборки операционных систем – Ubuntu Server, MacOS. Так же требуется поддержка следующих основных сервисов:

* сервер «Nginx»;
* сервер «Unicorn»;
* сервер баз данных «PostgreSQL».

Компьютер должен быть подключен к сети Интернет по широкополосному каналу. Осуществить это надо по возможности с использованием межсетевого экрана. Программную реализацию требуется создать на языках RUBY, HTML, SQL, Javascript с использованием фреймворка Rails и JavaScript-фреймворка AngularJS. Обязательно использование классов, функций и процедур, вместо повторения кода. Базы данных необходимо реализовывать посредством расширения драйвера [реляционных СУБД](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) postgresql, предоставляемого фреймворком Rails.

Таким образом, необходимо разработать веб-приложение, которое позволяло бы обычному пользователю управлять контентом сайта, добавлять, обновлять, удалять и динамически отображать информацию на страницах сайта. Готовое приложение должно быть интерактивным, удобным в использовании, иметь хорошо проработанную архитектуру и пользовательский интерфейс.

1. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРОГРАММНОГО СРЕДСТВА

**3.1 Общая структура программного средства**

Веб-приложение построено по REST архитектуре, в качестве backend части выступает Ruby on Rails приложение, которое взаимодействует с frontend частью – AngularJS приложение - посредством HTTP запросов и обмена данными в формате JSON. Структура программного средства изображена на рис. 3.1.

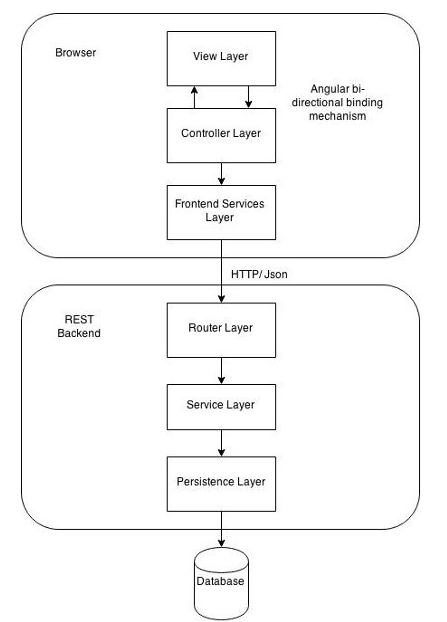


Рисунок 3.1 – Структура веб-приложения

REST (Representational state transfer) – это стиль архитектуры программного обеспечения для распределенных систем, таких как World Wide Web, который, как правило, используется для построения веб-служб. Термин REST был введен в 2000 году Роем Филдингом, одним из авторов HTTP-протокола. Системы, поддерживающие REST, называются RESTful-системами. В общем случае REST является очень простым интерфейсом управления информацией без использования каких-то дополнительных внутренних прослоек. Каждая единица информации однозначно определяется глобальным идентификатором, таким как URL. Каждая URL в свою очередь имеет строго заданный формат [13].

Архитектура в стиле REST состоит из клиентов и серверов. Клиенты инициируют запросы к серверам; серверы обрабатывают запросы и возвращают подходящие ответы. Запросы и ответы создаются на базе передачи представлений ресурсов. REST изначально описан в контексте HTTP, но не ограничен этим протоколом. Архитектуры типа RESTful могут быть основаны на других протоколах прикладного уровня, если они уже реализуют обширный и единый словарь для приложений, основанных на передаче значимых представлений состояний. Приложения RESTful увеличивают использование уже существующих хорошо определенных интерфейсов и других встроенных возможностей, предлагаемых выбранным сетевым протоколом, а также сокращают добавление к нему новых возможностей, специфичных для приложения.

Преимущества REST:

* масштабируемость взаимодействия компонентов;
* общность интерфейсов;
* независимое внедрение компонентов;
* промежуточные компоненты, снижающие задержку, усиливающие безопасность и инкапсулирующие устаревшие системы;

**3.2 Разработка модели базы данных**

Неотъемлемой частью конечного программного средства является база данных, используемая в процессе работы для хранения данных о пользователе и настройках приложения в разрабатываемом ПС. Поэтому целесообразно провести разработку информационной модели предметной области в виде схемы базы данных.

Процесс проектирования базы данных состоит из нескольких частей: разработка структуры базы данных и проведение процесса нормализации. Нормализация − это процесс преобразования отношений базы данных к виду, отвечающему нормальным формам. Нормализация предназначена для приведения структуры БД к виду, обеспечивающему минимальную логическую избыточность, и не предназначена для уменьшения или увеличения производительности работы или физического объёма базы данных.

PostgreSQL – это полноценная SQL СУБД с большим списком возможностей и огромным количеством людей по всему миру, которые используют и разрабатывают эту СУБД. В отличие от ещё одной свободной СУБД: MySQL, разработка которой изначально ориентировалась на веб, разработка PostgreSQL ориентировалась на использование в сложных приложениях. Именно поэтому упор всегда делался на надёжность, наличие развитой функциональности и соответствение стандартам. При этом, конечно, PostgreSQL можно точно также использовать и в веб-приложениях, где данная СУБД показывает неизменно отличные результаты, при лучшей масштабируемости и настраиваемости. PostgreSQL - это объектно-реляционная система управления базами основанная на POSTGRES, Версии 4.2, которая была разработана в Научном Компьютерном Департаменте Беркли Калифорнийского Университета. Слово "объект-реляционная" означает возможнсоть использьзования части функциональности, связанной с объектно-ориентированным программированием, например, такой как наследование. PostgreSQL является прямым потомком с открытым исходным кодом от оригинального кода, сделанного в Беркли. PostgreSQL поставляется под [лицензией BSD](http://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F_BSD), что обеспечивает максимальную открытость и доступность. СУБД является кроссплатформенным продуктом и работает не только в широком диапазоне диалектов UNIX (Linux, FreeBSD, Solaris и т.д.), но и на платформе Microsoft Windows.

Схема базы данных веб-приложения состоит из восемнадцати таблиц: таблицы «Users», «Roles», «Videos», «Votes», «Courses», «Categories», «Subscriptions», «Images», «Tasks», «Comments», «Mailboxer\_receipts», «Mailboxer\_conversations», «Mailboxer\_notifications», «Mailboxer\_conversati-on\_opt\_outs», «Admins» хранят данные об основных сущностях приложения; «User\_Roles», «Courses\_Subscriptions», «Categories\_Courses» необходимы для связи отношением «многие ко многим» основных сущностей.

Рассмотрим подробнее приведенную схему базы данных.

Таблица Users содержит данные о пользователе приложения. Состоит из полей:

1. provider – тип авторизации;
2. encrypted password – пароль в зашифрованном виде;
3. reset\_password\_token – токен смены пароля;
4. reset\_password\_sent\_at – время отправки сообщения о переустановке пароля;
5. remember\_created\_at – время создания напоминания пароля;
6. sign\_in\_count – количество попыток входа;
7. current\_sign\_in\_at – время начала сессии текущего пользователя;
8. last\_sign\_in\_at – время последней успешной авторизации;
9. current\_sign\_in\_ip – текущий ip-адресс пользователя;
10. last\_sign\_in\_ip – ip-адресс последней удачной сесии;
11. confirmation\_token – токен подтверждения пароля;
12. confirmed\_at – время подтверждения пароля;
13. confirmation\_sent\_at – время отправки подтверждения пароля;
14. unconfirmed\_email – email, который еще не подтвержден;
15. name – имя пользоваетля;
16. nickname – никнейм пользователя;
17. image – аватар пользователя;
18. email – текущий email пользователя;
19. tokens – токены аутентификации пользователя;
20. ban – поле блокировки пользователя.

Таблица Comments предназначена для хранения информации о комментариях пользователей. Связана с таблицей Users связью «один ко многим». Содержит следующую информацию:

1. body – содержит текст комментария;
2. commentable\_id – идентификатор сущности к которой относится комментарий;
3. commentable\_type – тип сущности к которой относится комментарий;
4. root\_comment\_id – идентификатор родительского комментария;
5. user\_id – связывает комментарий и пользователя.

Таблица Videos предназначена для хранения информации о видеолекциях пользователей. Связана с таблицей Course связью «один ко многим». Содержит следующую информацию:

1) video – название файла, содержащего видео;

1. name – название видеолекции;
2. format – формат видео;
3. description – описание видеолекции;
4. course\_id – связывает видеолекцию и учебный курс.

Таблица Votes предназначена для хранения информации о голосах пользователей. Связана с таблицей Users связью «один ко многим». Содержит следующую информацию:

1. vote\_flag – определяет тип голоса («понравилось», «не понравилось»);
2. votable\_id – идентификатор сущности к которой относится голос;
3. votable\_type – тип сущности к которой относится голос;
4. user\_id – связывает голос и пользователя.

Таблица Courses предназначена для хранения информации об учебных курсах. Связана с таблицей Categories связью «многие ко многим» через таблицу Categories\_Courses и с таблицей Users связью «один ко многим». Содержит следующую информацию:

1. name – содержит название учебного курса;
2. heading – краткое название курса (заголовок);
3. description – содержит описание учебного курса;
4. user\_id – связывает учебный курс и пользователя.

Таблица Roles предназначена для хранения информации о пользовательских ролях. Связана с таблицей Users связью «многие ко многим» через таблицу Users\_Roles. Содержит следующую информацию:

1) name – название роли;

1. resource\_id – иденцификатор сущности к которой относится роль;
2. resource\_type – тип сущности к которой относится роль.

Таблица Subscriptions предназначена для хранения информации о подписаных на учебный курс пользователях. Связана с таблицей Users связью «один ко многим». Содержит следующую информацию:

1. name – название подписки;

2) user\_id – связывает подписку на учебный курс и пользователя.

Таблица Categories предназначена для хранения информации о категориях учебных курсов. Связана с таблицей Courses связью «многие ко многим» через таблицу Categories\_Courses. Содержит следующую информацию:

1. name – содержит название категории учебных курсов.

Таблица Tasks предназначена для хранения дополнительных материалов учебных курсов. Связана с таблицей Courses связью «один ко многим». Содержит следующую информацию:

1. file – название файла, cодержащего задание;
2. format – формат файла;
3. name – название задания;

4) course\_id – связывает дополнительный материал и учебный курс.

Таблица Images предназначена для хранения информации о картинках. Связана с таблицей Videos связью «один ко многим» и с таблицей Courses связью «один ко многим». Содержит следующую информацию:

1. file – название файла;

2) course\_id – связывает картинку и учебный курс;

3) video\_id – связывает картинку и видеолекцию.

Связь объектов и базы данных в веб-приложении осуществляется с помощью технологии Object-Relational Mapping, которая называется [ActiveRecord](http://wiki.rubyonrails.org/rails/pages/ActiveRecord). Object-Relational Mapping - технология программирования, которая связывает [базы данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)  с моделями [объектно-ориентированных языков программирования](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и занимается отображением полей из таблицы базы данных в поля объекта. Кроме этого, технология отвечает за валидацию объектов перед сохранением и генерацию кода для представления связей между объектами. Модель создается путем наследования от класса ActiveRecord::Base :

class Course < ActiveRecord::Base

belongs\_to :user

has\_many :videos, dependent: :destroy

end

Данный код создает модель Course и связывает ее с таблицей Courses в базе данных.

Использование ActiveRecord также предоставляет возможность воспользоваться сгенерироваными CRUD методами для модели Course:

* Course.find(id) – найти курс по id (PrimaryKey в БД, обычно это Integer);
* Course.find(:all) – выбрать все курс;
* Course.find\_by\_name('C++!') – найти курс с заголовком “Hello World!”;
* Course.create(name: 'Ruby!') – создать курс и сохранить в БД;
* Course.update(name: 'Python!') – обновить курс в БД;
* Course.destroy – удалить курс из БД.

Для управления схемой базы данных фреймворк Rails использует механизм миграций. Миграции хранятся в файлах, запускаемых для любой базы данных, которую поддерживает Active Record, с использованием утилиты rake. Ниже представлена миграция, создающая таблицу Courses:

class Course < ActiveRecord::Migration

def change

create\_table :courses do |t|

t.string :name , :null => false, :default => ""

t.string :heading

t.text :description

t.belongs\_to :user, index: true

t.timestamps null: false

end

end

end

Rails отслеживает, какие файлы переданы в базу данных, и представляет возможность отката. Вся информация о подключении и настройках базы данных находится в файле database.yml.

* 1. **Программная реализация**

**3.3.1 Rest API back-end**

В качестве веб-сервиса или API выступает Ruby on Rails приложение, состоящее из моделей, контроллеров и сериализаторов.

Модель предоставляет остальным компонентам приложения объектно-ориентированное отображение данных (таких как учебный курс или комментарий). Объекты модели могут осуществлять загрузку и сохранение данных в реляционной базе данных, а также реализуют бизнес-логику. Для хранения объектов модели в СУБД PostgreSQL по умолчанию в Rails 4 использована библиотека ActiveRecord.

Данное rails приложение состоит из моделей: Role, User, Category, Subscription, Message, Course, Task, Video, Comment, Image, Vote, Receipt.

Контроллер в Rails — это набор логики, запускаемой после получения HTTP-запроса сервером. Контроллер отвечает за вызов методов модели и запускает формирование представления, но в данном приложении вместо формирования представления формируется JSON объект. Соответствие интернет-адреса с действием контроллера (маршрут) задается в файле config/routes.rb. Открытые методы контроллера являются так называемыми действиями (actions). Action часто соответствует отдельному представлению либо объекту JSON.

API часть приложения состоит из следующих контроллеров: UsersController, VideoController, SubscriptionsController, CommentsController, TasksController, CategoriesController, CoursesController, MessagesController, ConversationsController. Все контроллеры веб-приложения наследуются от ApplicationController.

Сериализатор – класс, который используется для сериализации объектов. Сериализация используется для передачи [объектов](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) по сети и для сохранения их в [файлы](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB). В данном приложении для создания сериализаторов используется класс ActiveModel: Serializer, который облегчает процесс формирования объектов JSON.

В backend части веб-приложения написаны следующие сериализаторы: CategoriesSerializer, CommentSerializer, ConversationSerializer, CourseSerializer, RoleSerializer, TasksSerializer, UsersSerializer, VideoNameSerializer, VideoSerializer, VoteVideoSerializer.

class CommentSerializer < ActiveModel::Serializer

attributes :id, :body, :user\_id, :root\_comment\_id, :created\_at, :sender\_nickname, :root\_comment\_nickname

def sender\_nickname

object.user.nickname

end

def root\_comment\_nickname

object.get\_root\_comment\_nickname(object.root\_comment\_id) if object.root\_comment\_id

end

end

**3.3.2 Клиентская часть**

Клиентская часть приложения выстраиваться вокруг модели, относящейся к представлению. Здесь обрабатывается только логика представления. В клиентской части выделяется три уровня: представления, управления и клиентских сервисов.

Уровень представления состоит из HTML-шаблонов, таблиц CSS и директив Angular, соответствующих различным компонентам пользовательского интерфейса. Представление для страницы отображения учебных курсов:

<div id="courses" class="container">

<div class="row">

<div class="col-xs-4">

<label>Поиск:</label>

<input type="text" class="form-control" placeholder="Название курса" ng-model="query">

</div>

<div class="col-xs-4">

<label>Сортировать по:</label>

<select class="form-control" ng-init="sortBy = 'none'" ng-model="sortBy">

<option value="none">-- Выбор --</option>

<option value="name">Название</option>

<option value="created\_at">Дата</option>

</select>

</div>

<div class="col-xs-4">

<label>Выбор категории:</label>

<select class="form-control" ng-model="courses.selectedCat">

<option value="">Все</option>

<option ng-repeat="cat in courses.categories">{{cat.name}}</option>

</select>

</div>

</div>

<div class="row from-filters">

<div class="col-xs-4" ng-repeat=" item in courses.courses | filter:

{name: query, categories: { name: courses.selectedCat }} | orderBy:sortBy">

<a ui-sref="applicationLayout.videos({id: {{ item.id }} })" class="link\_to\_course">

<div class="panel panel-default">

<div class="panel-heading title">

{{ item.name }}

</div>

<div class="panel-body">

<img class="content" ng-src = {{item.image}} />

</div>

<div class="panel-body" ng-repeat="cat in item.categories" style="float:left">

<a href="javascript:void(0)" ng-click="courses.selectedCat = cat.name" class="fa fa-tags">{{cat.name}}</a>

</div>

</div>

</a>

</div>

</div>

</div>

Уровень управления (контроллера) состоит из контроллеров Angular, склеивающих вместе данные, извлекаемые, соответственно, из машинного интерфейса и из представления. Контроллер инициализирует модель представления и определяет, как представление должно реагировать на изменения модели и наоборот. Контроллер соответствующий модели представления учебных курсов:

.controller('CoursesCtrl', CoursesCtrl);

CoursesCtrl.$inject = [

'ENV', '$scope', '$state', 'Course', 'Category'

];

function CoursesCtrl(ENV, $scope, $state, Course, Category) {

console.log('CoursesCtrl load');

var vm = this;

vm.selectedCat = undefined;

vm.getCourses = getCourses;

vm.getCategories = getCategories;

vm.courses = undefined;

vm.categories = undefined;

getCourses();

function getCourses(){

Course.get({all: true}).$promise

.then(function(data) {

vm.courses = data.courses;

getCategories();

})

.catch(function() {

console.log('courses load error');

});

}

function getCategories(){

Category.get().$promise

.then(function(data) {

vm.categories = data.categories;

})

.catch(function() {

console.log('categories load error');

});

}

}

Одна из основных задач контроллера — выполнять валидацию в клиентской части. Все подобные клиентские валидации предусматриваются лишь для удобства пользователя — например, с их помощью удобно немедленно информировать пользователя о том, что поле обязательно для заполнения. Все акты клиентской валидации повторяются на машинном интерфейсе (на уровне сервисов) из соображений безопасности, поскольку клиентскую валидацию легко обойти.

Уровень клиентских сервисов представлен фабриками, которые взаимодействуют с машинным интерфейсом. Фабрика учебного курса дает возможность получать данные об учебном курсе и удалять учебный курс по id:

.factory('Course', ['$resource', 'ENV', function($resource, ENV) {

return $resource(ENV.apiEndpoint + "/courses/:id",

{

id: '@id'

},{

charge: {method:'DELETE', params:{id: '@id'}}

});

}])

**3.4 Тестирование**

**3.4.1 Модульное тестирование**

Модульное тестирование (юнит-тестирование, unit testing) – это процесс в программировании, который позволяет проверить на корректность конкретные модули исходного кода. Суть заключается в том, чтобы писать тесты для каждого нетривиального метода или функции, что позволит очень быстро проверить, не привело ли какое-либо изменение исходного кода к появлению ошибок в уже оттестированных частях программы. Так же тестирование облегчает обнаружение у устранение таких ошибок. Цель юнит-тестирования – это изоляция отдельных частей программы и демонстрация того, что по отдельности эти части работоспособны. Данный тип тестирования в основном выполняется программистами.

Преимущества:

* модульное тестирование позволяет проводить рефакторинг, будучи уверенным, что модуль корректно работает;
* юнит-тестирование также помогает устранить сомнения по поводу отдельных модулей. Также может быть использован сначала для тестирования отдельных частей, а затем программы в целом;
* модульный тест можно рассматривать как «живой документ» для тестируемого класса [14].

Как и любое тестирование, юнит-тестирование не позволяет выявить все ошибки программы. Это следует из невозможности трассировки всех возможных путей выполнения программы. Помимо этого, каждый модуль тестируется отдельно, а это значит, что ошибки интеграции системного уровня, функций, которые исполняются в нескольких модулях, определены не будут. Технология также бесполезна при тестировании на производительность. Поэтому разумно будет использовать модульное тестирование в сочетании с другими методиками.

Для модульного тестирования веб-приложения использовались Rspec и Factory\_girl.

RSpec является DSL(Domain Specific Language), то есть языком, который предназначен для какой-то конкретной области. Написан на Ruby и применяется для тестирования приложений на Ruby. Его можно использовать, включив в проект библиотеку rspec, которая является пакетом, содержащим три подбиблиотеки: rspec-core, rspec-expectations и rspec-mocks. RSpec настроен так, что при включении его в проект, есть возможность запустить тесты из командной строки.

Все контроллеры и модели back-end части веб-приложения покрыты модульными тестами rspec. Код тестов для модели видеолекции:

require 'rails\_helper'

RSpec.describe Video, type: :model do

describe '#full\_url' do

let(:video) {FactoryGirl.create(:video)}

it 'should be exist' do

expect(video.full\_url.present?).to eq(true)

end

it 'should return host and port' do

expect(video.full\_url.

should include ('http://localhost:3000')).to eq(true)

end

it 'should return video' do

expect(video.full\_url.should include ('vd.mp4')).to eq(true)

end

end

end

Factory\_girl используется для удобного создания экземпляров моделей. С помощью Factory\_girl достаточно один раз определить фабрику (шаблон) и далее генерировать новые сущности на его основе. В связке с RSpec  фабрики хранятся в папке spec/factories. Код фабрики видеолекции:

FactoryGirl.define do

factory :video do

name 'XVS'

format 'mp4'

description 'New video.'

video File.open(File.join(Rails.root, 'spec', 'support', 'files', 'vd.mp4'))

factory :video\_with\_image do

after(:create) do |video|

video.image = FactoryGirl.build(:image)

video.save

end

end

end

end

**3.4.2 Функциональное тестирование**

Функциональное тестирование является одним из ключевых видов тестирования, задача которого – установить соответствие разработанного программного обеспечения (ПО) исходным функциональным требованиям заказчика. То есть проведение функционального тестирования позволяет проверить способность информационной системы в определенных условиях решать задачи, нужные пользователям.

В зависимости от степени доступа к коду системы можно выделить два типа функциональных испытаний:

* тестирование black box (черный ящик) – проведение функционального тестирования без доступа к коду системы;
* тестирование white box (белый ящик) – функциональное тестирование с доступом к коду системы.

Тестирование black box проводится без знания внутренних механизмов работы системы и опирается на внешние проявления ее работы. При этом тестировании проверяется поведение ПО при различных входных данных и внутреннем состоянии систем. В случае тестирования white box создаются тест-кейсы, основанные преимущественно на коде системы ПО. Также существует расширенный тип black-box тестирования, включающего в себя изучение кода, – так называемый grey box (серый ящик).

Преимущества:

* функциональное тестирование ПО полностью имитирует фактическое использование системы;
* позволяет своевременно выявить системные ошибки ПО и, тем самым, избежать множества проблем при работе с ним в дальнейшем;
* экономия за счет исправления ошибок на более раннем этапе жизненного цикла ПО;
* функциональное тестирование веб приложения проводилось на компьютере с операционной системой Windows 10 в браузере Chrome [15].

Таблица 3.1 – Набор тест-кейсов модуля «Авторизации» и «Регистрации»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тест-кейса | Тестируемая функциональность | Последовательность действий | Ожидаемый  результат | Полученный результат |
| 1 | Авторизация. Валидация ошибок | 1. Запустить приложение.  2.Нажать на кнопку «Войти». | 1. Сообщение под полем ввода логина «Пожалуйста введите Ваш email адресс». | Тест успешно пройден |
| 2 | Авторизация. Валидация ошибок | 1. Запустить приложение.  2. Ввести заведомо ложный логин и пароль.  3.Нажать на кнопку «Войти». | 1. Сообщение под полевом ввода логина «Пожалуйста введите корректный email и пароль». | Тест успешно пройден |
| 3 | Регистрация. Валидация ошибок. | 1. Запустить приложение.  2. Перейти на страницу регистрации  3.Заполнить все поля и пароль менее 8 символов  4. Нажать кнопку «Зарегистрировать» | 1. Сообщение под полевом ввода пароля «Пароль должен содержать не менее 8 символов ». | Тест успешно пройден |
| 4 | Регистрация. Успешная регистрация. | 1. Запустить приложение.  2. Перейти на страницу регистрации  3.Заполнить все поля корректно.  4. Нажать кнопку «Зарегистрироваться» | 1. Сообщение с текстом «Регистрация прошла успешно». | Тест успешно пройден |

Следующий набор тест-кейсы рассматривается под ролью «Слушатель». Шаг ввода корректного пароля и логина выполнен.

Таблица 3.2 – Набор тест-кейсов модуля «Учебный курс»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тест-кейса | Тестируемая функциональность | Последователь-ность действий | Ожидаемый  результат | Полученный результат |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Отображение учебных курсов с учетом выбранной категории. | 1. Выбрать категорию учебных курсов для отображения данных | 1. Отображение информации об учебных курсах, относящихся к заданной категории. | Тест успешно пройден |
| 2 | Отображение учебных курсов с учетом введенного названия для поиска. | 1. Ввести название учебного курса для отображения данных | 1. Отображение информации об учебных курсах с заданным названием. | Тест успешно пройден |
| 3 | Оформление подписки на учебный курс. Успешная подписка. | 1. Переход на страницу выбранного учебного курса.  2. Кликнуть по кнопке «Подписаться» в правом верхнем углу экрана. | 1. Отображение информации о выбранном учебном курсе.  2. Сообщение с текстом «Вы успешно подписаны на данный учебный курс». | Тест успешно пройден |
| 4 | Оформление подписки на учебный курс. Валидация ошибок. | 1. Переход на страницу учебного курса, на который пользователь уже подписан. | 1. Отображение информации о выбранном учебном курсе.  2. Сообщение с текстом «Вы уже подписаны на данный учебный курс». | Тест успешно пройден |
| 5 | Отображение видеолекций выбранного учебного курса | 1. Переход на страницу выбранного учебного курса. | 1. Отображение информации о видеолекциях выбранного учебного курса. | Тест успешно пройден |

Продолжение таблицы 3.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 6 | Оценка видеолекции | 1. Переход на страницу выбранного учебного курса.  2. Переход на страницу просмотра видеолекции  3.Клик по кнопке «понравилось» либо «не понравилось» | 1. Отображение информации о видеолекциях выбранного учебного курса.  2. Отображение видео выбранной видеолекции  3.Увеличение голосов «понравилось» («не понравилось») | Тест успешно пройден |
| 7 | Комментирование видеолекции | 1. Переход на страницу выбранного учебного курса.  2. Переход на страницу просмотра видеолекции  3.Заполнение формы комментария видеолекции. Клик по кнопке «Добавить». | 1. Отображение информации о видеолекциях выбранного учебного курса.  2. Отображение видео выбранной видеолекции  3.Отображение комментария пользователя под видеолекцией. | Тест успешно пройден |

Следующий набор тест-кейсы рассматривается под ролью «Лектор». Шаг ввода корректного пароля и логина выполнен.

Таблица 3.3 – Набор тест-кейсов модуля «Учебный курс»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тест-кейса | Тестируемая функциональность | Последовательность действий | Ожидаемый  результат | Полученный результат |
| 1 | 2 | 3 | 4 | 5 |
| 8 | Успешное добавление нового учебного курса | 1. Переход на страницу «Мои курсы».  2.Клик по кнопке «добавить».  3.Правильное заполнение формы добавления курса.Клик по кнопке «Добавить» | 1. Отображение информации о моих учебных курсах.  2. Отображение формы добавления учебного курса  3. Сообщение с текстом «Ваш учебный курс успешно добавлен». | Тест успешно пройден |

Продолжение таблицы 3.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 9 | Добавление учебного курса.  Валидация ошибок. | 1. Переход на страницу «Мои курсы».  2.Клик по кнопке «Добавить».  3. Не заполнять любое поле, окрашеное в красный цвет | 1. Отображение информации о моих учебных курсах.  2. Отображение формы добавления учебного курса  3. Сообщение с текстом «Необходимо заполнить все обязательные поля». | Тест успешно пройден |
| 10 | Удаление учебного курса | 1. Переход на страницу «Мои курсы».  2. Подвести курсор мыши к учебному курсу, который необхордимо удалить  3. Клик по кнопке «Удалить» | 1. Отображение информации о моих учебных курсах.  2. Отображение кнопки удаления для учебного курса.  3. Сообщение с текстом «Учебный курс удален успешно». | Тест успешно пройден |

Успешность прохождения тестов показывает корректность работы программы с реальными данными и соответствие функциональным требованиям.

**4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

* 1. **Пользовательская роль «Гость»**

После того как пользователь попадает на главную страницу приложения, ему будет предложено пройти процесс аутентификации, используя свой логин и пароль. Если пользователь не зарегистрирован в системе, он может пройти процедуру регистрации, перейдя по ссылке «регистрация». Форма регистрации представлена на рис. 4.1:

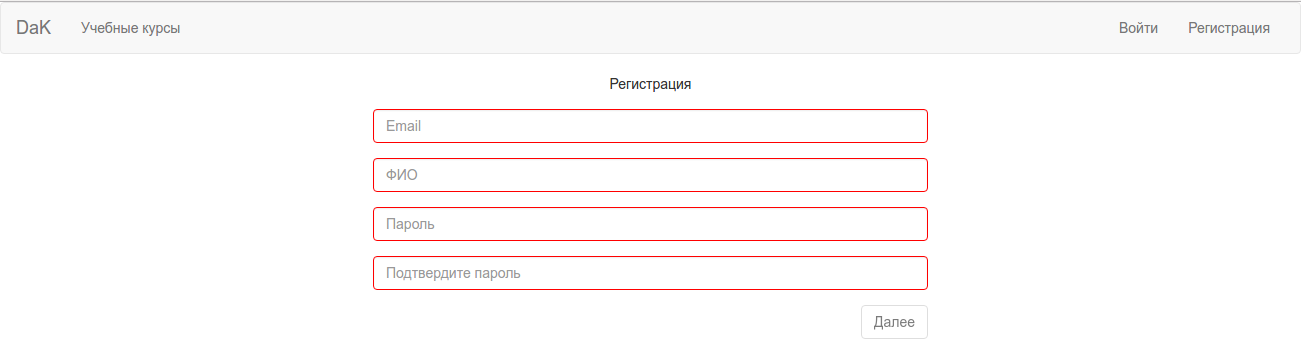


Рисунок 4.1 – Форма регистрации

После успешных процедур регистрации и аутентификации, пользователь попадает на домашнюю страницу приложения.

Домашняя страница приложения изображена на рис. 4.2 и представляет из себя окно, с представленными учебными курсами и возможностью поиска по ним.

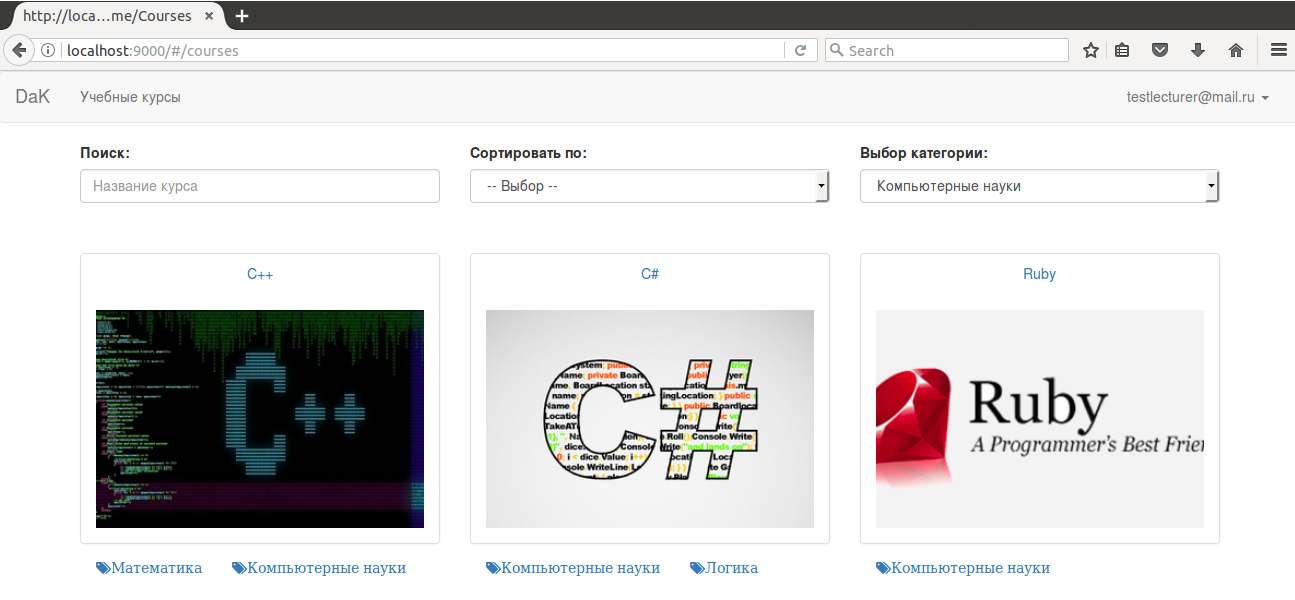


Рисунок 4.2 – Домашняя страница

Для просмотра своего статуса пользователю необходимо перейти в меню редактирование личных данных, которое расположено в правом верхнем углу экрана. Если пользователь только что прошел процедуру регистрации, то его статус будет помечен как «зарегистрированный пользователь».

Для пользователя с ролью «Зарегистрированный пользователь» доступна только домашняя страница приложения, возможность редактирования личных данных, получение и отправка сообщений.

* 1. **Пользовательская роль «Слушатель»**

Пользователь с ролью «Слушатель», помимо функций «Зарегистрированного пользователя», имеет доступ к основной функциональности веб-приложения. По нажатию на выбранный учебный курс слушатель переходит на страницу учебного курса в соответствии с рис. 4.3.

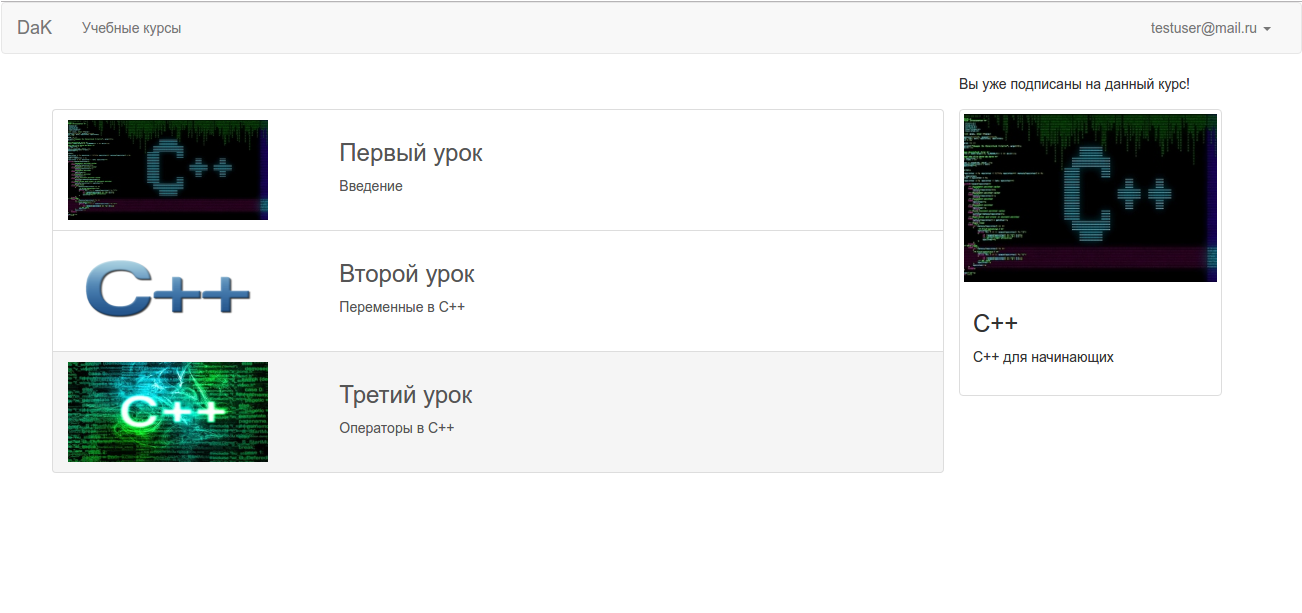


Рисунок 4.3 – Страница учебного курса

На данной странице, по нажатию на кнопку «Оформить подписку», слушатель может оформить подписку на учебный курс, и, в последующем, данный курс будет отображаться в меню «Мои учебные курсы».

К каждому учебному курсу лектор добавляет необходимый дополнительный материал, который доступен для скачивания слушателю под логотипом курса.

Учебный курс состоит из видеолекций, на которые слушатель может перейти кликнув по ней левой кнопкой мыши. На странице видеолекции, которая изображена на рис. 4.4 слушатель может посмотреть видео, оценить его, оставить свой комментарий.

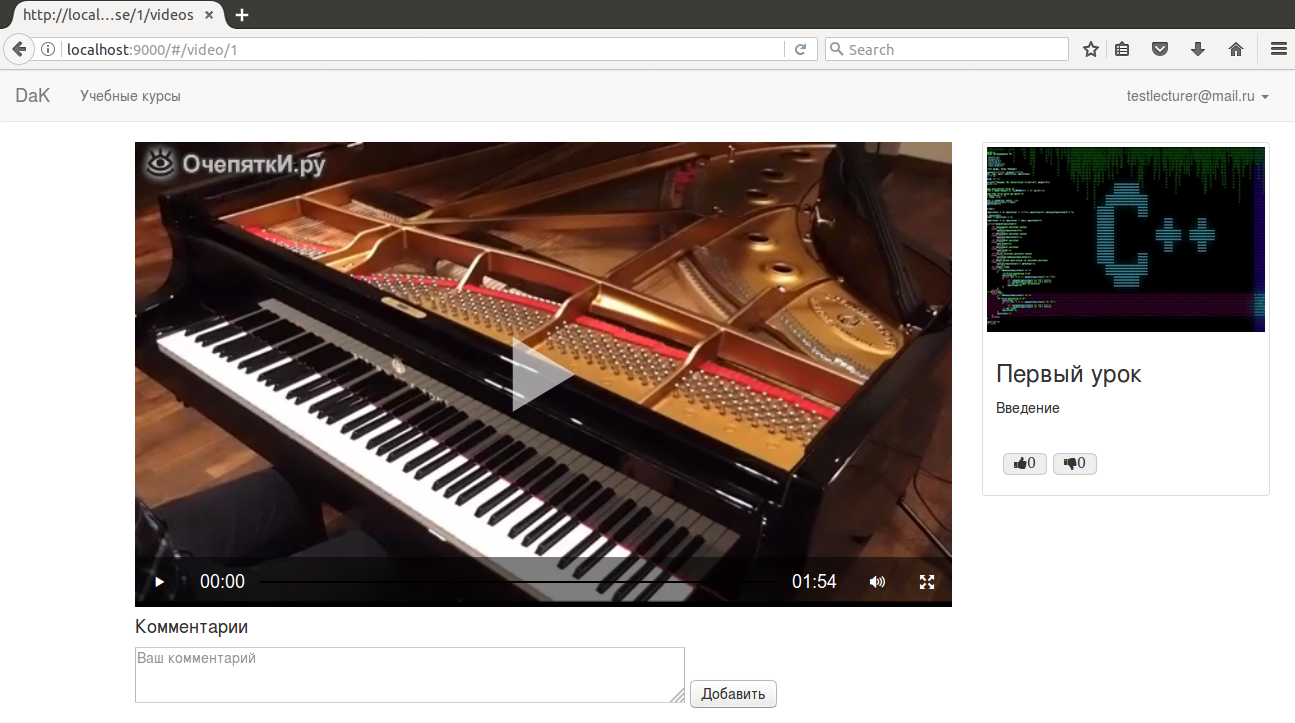


Рисунок 4.4 – Страница видеолекции

Для просмотра видео на страницу видеолекции встроен видеоплеер. Управление видеоплеера стандартное: для проигрывания видео необходимо нажать кнопку «Play», для остановки «Stop», также присутствует возможность перемотки и регулировки звука. Для того, чтобы развернуть видео на весь экран необходимо кликнуть левой кнопкой мыши по кнопке видеоплеера расположенной в правом нижнем углу, для возврата нажать кнопку «Esc».

Для оценки видеолекции пользователю необходимо нажать кнопку «Понравилось» либо «Не понравилось». При попытке проголосовать за видеолекцию повторно, голос пользователя будет снят.

Для добавления комментария необходимо ввести текст в поле ввода под видеоплеером и нажать кнопку «Добавить».

* 1. **Пользовательская роль «Лектор»**

Основное назначение пользователя с ролью лектор – добавление учебных курсов, управление ими, а также коммуникация со слушателями, которым интересен учебный курс данного лектора.

Для создания учебного курса, пользователю с данной ролью необходимо перейти на вкладку «Мои курсы» и, после отображения всех учебных курсов данного лектора, нажать кнопку «Добавить учебный курс».

Форма добавления учебного курса изображена на рис. 4.5 и представляет собой набор полей, красным выделены поля обязательные для заполнения пользователем. После их правильного заполнения под формой активируется кнопка добавить, после клика по которой лектор добавит свой учебный курс.

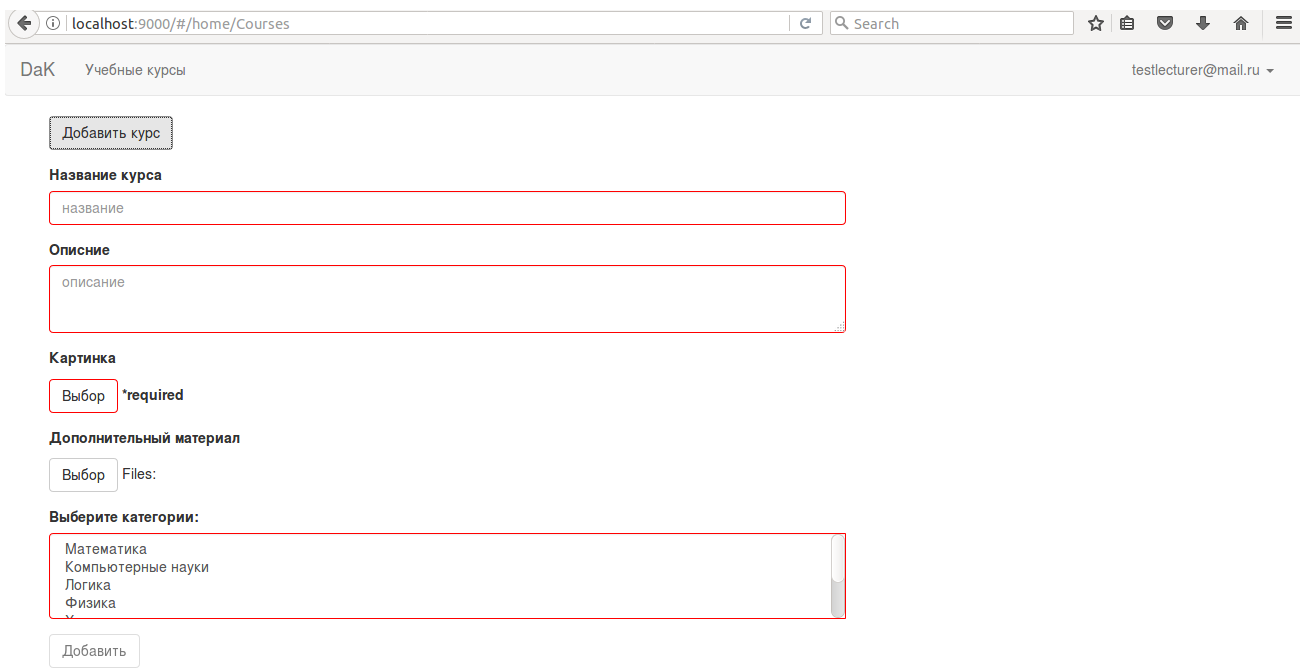


Рисунок 4.5 - Страница добавления учебного курса

После перехода на страницу учебного курса, которая изображена на рис. 4.6, лектор может добавить файлы дополнительного материала, удалить виделекции или добавить новые.

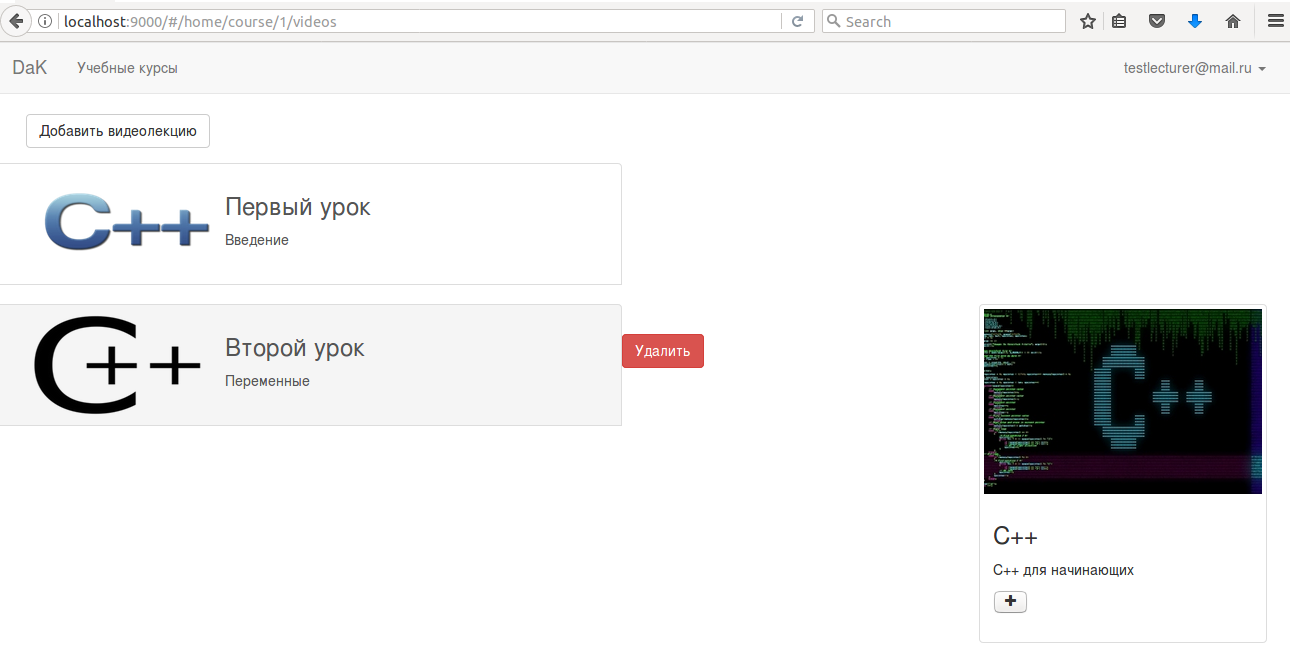


Рисунок 4.6 - Страница учебного курса

Для добавления дополнительного материала необходимо нажать кнокпу «+» под логотипом курса, и выбрать файлы. После этого под перечнем имен файлов кликнуть по кнопке «Прикрепить».

Для добавления видеолекции лектору необходимо кликнуть по кнопке «Добавить видеолекцию», после появления формы добавления заполнить ее корректно и кликнуть по кнопке «Добавить». При наведении курсора мыши на уже добавленную лекцию, справа от логотипа всплывает кнопка «Удалить» с помощью которой происходит удаление видеолекции.

**4.4 Пользовательская роль «Администратор»**

Администратор – пользователь, который имеет доступ ко всем страницам приложения. Основное назначение данного пользователя – управление ролями и категориями. Страница администратора состоит из двух вкладок: пользователи и категории.

Вкладка «Пользователи» изображена на рис. 4.7 и представляет собой список. Рядом с каждым элементом данного списка расположен чекбокс, с помощью которого администратор может заблокировать выбранного пользователя, и ссылка на страницу редактирования пользователя. На странице редактирования администратор может посмотреть личный профайл пользователя и, при необходимости, изменить роль пользователю, используя выпадающее меню для назначения определенной роли.

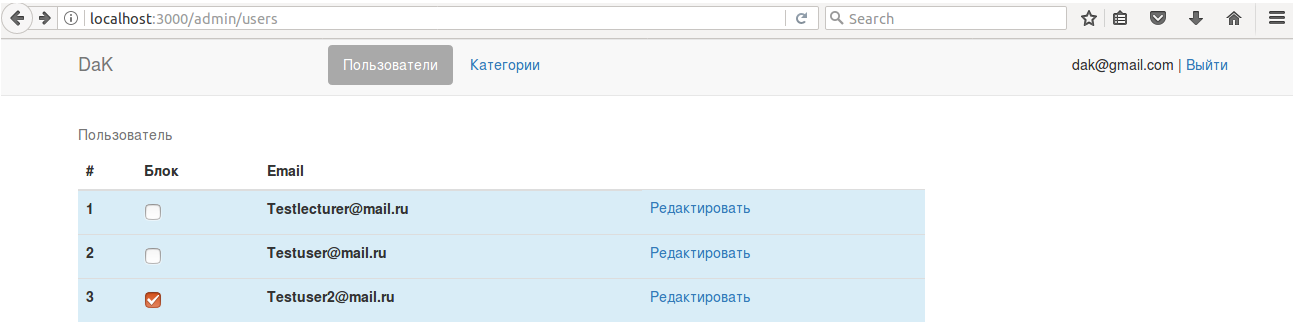


Рисунок 4.7 – Вкладка «Пользователи»

В соответствии с рис. 4.8 вкладка «Категории» служит для управления списком категорий учебных курсов.

Следует отметить, что пользователю с ролью «Администратор» открыт доступ ко всем страницам веб-приложения и данный пользователь может просматривать и редактировать комментарии всех пользователей и также удалять учебные курсы, которые не соответсвуют правилам и тематике ресурса.

Адиминистратор, как и любой другой пользователь, участвует в перепске посредством личных сообщений. Любой желающий может задать вопрос или отправить свои отзывы и пожелания по работе сайта личным сообщением администратору.

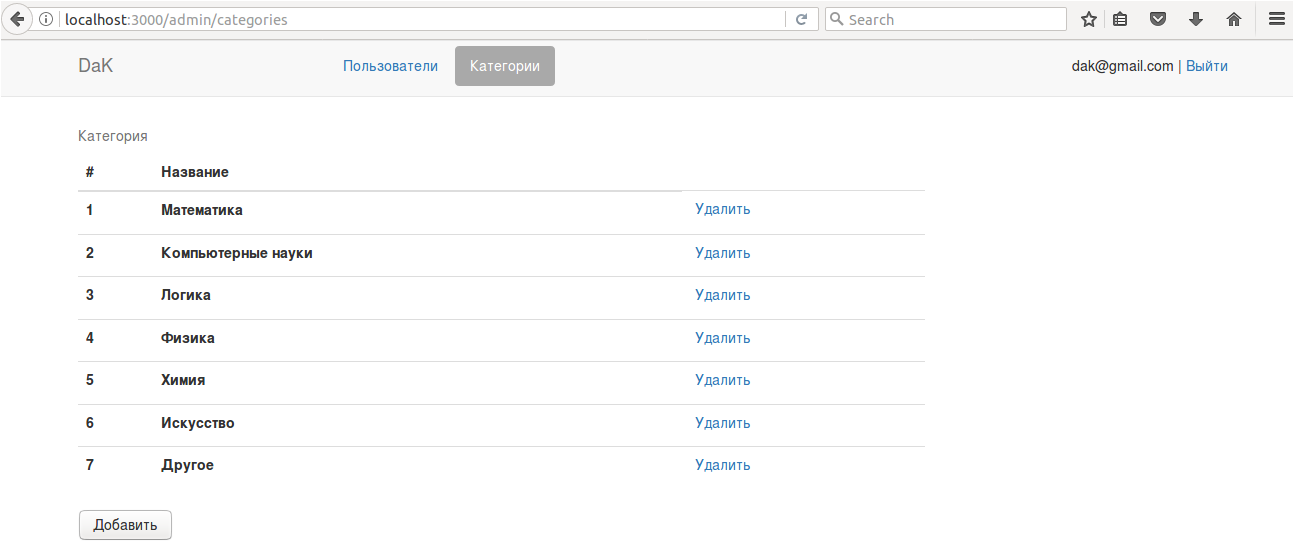


Рисунок 4.8 – Вкладка «Категории»

**5 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ И ИСПОЛЬЗОВАНИЯ ВЕБ-ПРИЛОЖЕНИЯ**

**5.1 Общая характеристика разрабатываемого программного средства**

Первым и одним из основополагающих этапов разработки любого программного продукта является обоснование необходимости появления данного программного продукта на рынке, его возможная востребованность, а также анализ экономической эффективности разработки. Выбор эффективных проектов основывается на их экономической оценке и расчете экономического эффекта. Поэтому разрабатываемые программные продукты должны быть не только совершенными в техническом плане, но и выгодными с точки зрения экономики.

Целью технико-экономического обоснования эффективности разработки программного продукта является определение экономической выгодности создания данного продукта и его дальнейшего применения.

Разработка веб-приложения для онлайн обучения с поддержкой мультимедийного контента позволит любому желающему получить необходимые знание в той или иной сфере, усовершенствовать процесс обучения в учебных заведениях Республики Беларусь.

Использование данного программного средства позволит значительно снизить затраты времени как преподавателей, так и слушателей. Студенты будут иметь доступ к нужной им информации в любое время суток, а также смогут получить ответы на все возникшие вопросы от преподавателей не выходя из дома. Данное программное средство также позволит сократить затраты денежных средств на оплату труда преподавателей за счет информативности системы и возможного сокращения часов на лекции в будущем.

Целью экономического раздела является экономическое обоснование разработки программного средства, а также определение организационно-экономических условий ее эффективного функционирования.

По окончании расчетов будет сделан вывод об экономической целесообразности использования программного средства.

**5. 2 Смета затрат и цена программного обеспечения**

**5.2.1 Определение объема и трудоемкости ПО**

Экономический эффект от разработки программного продукта определяется чистой прибылью, остающейся в распоряжении организации-разработчика после реализации данного программного продукта.

С учетом того, что стадии проектирования определяются только для крупных проектов (более 3 исполнителей), то для данного проекта стадии не определяются.

Данный программный продукт относится к 3-й группе сложности. Категория новизны – «В».

Разработанное автором программное обеспечение позволяет производить регистрацию через социальные сети, а также ему присуща интерактивность. Таким образом, ему присущи 2 характеристики из перечисленных в таблице П.4.2, поэтому дополнительный коэффициент сложности ПО равен 0,12.

Исходные данные для проведения расчетов приведены в таблице 5.1.

Таблица 5.1 – Исходные данные

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Наименование показателя | Обозначение | Единица измерения | Количество |
| 1 | Коэффициент новизны | Кн | единиц | 0,7 |
| 2 | Дополнительный коэффициент сложности | Кс | единиц | 1,12 |
| 3 | Поправочный коэффициент, учитывающий использование типовых программ | Кт | единиц | 0,7 |
| 4 | Установленная плановая продолжительность разработки | Тр | лет | 0,25 |
| 5 | Продолжительность рабочего дня | Тч | часов | 8 |
| 6 | Тарифная ставка 1-го разряда | Тм1 | руб. | 31 |
| 7 | Коэффициент премирования | Кп | единиц | 1,8 |
| 8 | Норматив дополнительной заработной платы исполнителей | Нд | % | 25 |
| 9 | Отчисления в ФСЗН | Зсз | % | 34 |
| 10 | Отчисления в Белгосстрах | Нне | % | 0,6 |
| 11 | Прочие прямые расходы | Нпз | % | 20 |
| 12 | Уровень рентабельности ПО | Урпi | % | 40 |

Расчет чистой прибыли от реализации начинается с определения стоимостной оценки программного продукта. Базой для определения стоимостной оценки программного продукта является смета затрат, на основании которой определяется себестоимость и приемлемая цена реализации данного программного продукта. В свою очередь, для расчета плановой сметы затрат необходимо определить объем программного продукта.

Объем ПС определяется на основе данных, приведенных в таблице 5.2.

Таблица 5.2 – Характеристика и объем функций

|  |  |  |  |
| --- | --- | --- | --- |
| Номер функции | Наименование (содержание) функции | Объем функции | |
| LOC | |
| по аналогичным проектам | уточненный |
| (Vi) | (Vуi) |
| 101 | Организация ввода информации | 50 | 50 |
| 109 | Организация ввода-вывода информации в интерактивном режиме | 320 | 320 |
| 111 | Управление вводом/выводом | 250 | 320 |
| 203 | Формирование баз данных | 320 | 320 |
| 204 | Обработка наборов и записей базы данных | 240 | 240 |
| 206 | Обслуживание базы данных в интерактивном режиме | 260 | 260 |
| 207 | Манипулирование данными | 210 | 210 |
| 208 | Организация поиска и поиск в базе данных | 800 | 800 |
| 305 | Обработка файлов | 300 | 300 |
| 506 | Обработка ошибочных и сбойных ситуаций | 190 | 190 |
| 707 | Графический вывод результатов | 600 | 600 |
| Итого |  | 3540 | 3540 |

Среда разработки веб-приложения – RubyMine 8, [PostgreSQL](https://www.google.by/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwiF__zWi4PTAhUmIJoKHSXKBjIQFggaMAA&url=https%3A%2F%2Fwww.postgresql.org%2Fabout%2Fnews%2F1481%2F&usg=AFQjCNHHk821EpEXU2XZia_x0LWOH0mYQg&sig2=qMBYuiHEGuKB3RIctfeMkg) 9.3. Программный продукт относится к категории прикладного ПО.

Расчет объема программного продукта (количества строк исходного кода) предполагает определение типа программного обеспечения, всестороннее техническое обоснование функций ПО и определение объема каждой функции. На стадии технико-экономического обоснования проекта невозможно рассчитать точный объем функций. В нашем случае получены ориентировочные (прогнозные) оценки на основе имеющихся фактических данных по аналогичным проектам, которые были уточнены (скорректированы) с учетом условий разработки ПО в организации.

Общий объем функций программного средства вычисляется по формуле 5.1.

|  |  |
| --- | --- |
|  | (5.1) |

где – общий объем ПС;

– объем отдельной функции ПС;

– общее число функций.

Таким образом, общий объем ПС равен (строк исходного кода).

Уточненный объем функций программного средства вычисляется по формуле 5.2.

|  |  |
| --- | --- |
|  | (5.2) |

где – уточненный объем ПС;

– объем отдельной функции ПС;

– общее число функций.

Таким образом, общий объем ПС равен (строк исходного кода).

Затем по уточненному объему программного продукта и нормативам затрат труда в расчете на единицу объема определяются нормативная и общая трудоемкость разработки программного продукта.

Нормативная трудоемкость, которая определяется на основании принятого к расчету объема и категории сложности, а также уточняется с учетом сложности и новизны проекта и степени использования в процессе разработки стандартных модулей, является основой для расчета общей трудоемкости.

Общая трудоемкость программного продукта рассчитывается по следующей формуле:

, (5.3)

где Кс – коэффициент, учитывающий сложность ПО;

Кт – поправочный коэффициент, учитывающий степень использования при разработке стандартных модулей (согласно табл. 4.5 принимаем KТ = 0,8);

Кн – коэффициент, учитывающий степень новизны ПО (принимаем Kн=0,7);

Tн – нормативная трудоемкость (согласно прил. 3 при Vу= 3 540 и первой категории сложности Tн= 77).

Наличие двух характеристик по категории сложности позволяет применить к объему ПО коэффициент Kс:

Таким образом, общая трудоемкость разработки данного программного продукта составит:

Для эффективного распределения трудовых ресурсов организации-разработчика целесообразно по каждому проекту определить число исполнителей, необходимых для задействования на данном проекте. Численность исполнителей проекта можно определить по формуле:

|  |  |
| --- | --- |
|  | (5.4) |

где Тр – срок разработки проекта, лет;

То  – общая трудоемкость разработки проекта, чел/дн.;

Фэф – эффективный фонд времени работы одного работника в течении года, дн.

Срок разработки проекта – 3 месяца. Эффективный фонд времени определяется по формуле:

|  |  |
| --- | --- |
| , | (5.5) |

где Дк – количество календарных дней в году, дн.;

Дв – количество выходных дней в году, дн.;

Дп – количество праздничных дней в году, не совпадающих с выходными, дн.;

До – количество дней отпуска, дн.

Таким образом, для успешной разработки программного средства целесообразно задействовать 1 человека; общая трудоемкость программного продукта составит 42,63 чел/дн.

**5.2.2 Расчет сметы затрат и цены заказного ПО**

**5.2.2.1** **Расчет основной заработной платы**

В настоящее время основной статьей расходов на создание программных продуктов является заработная плата разработчиков, а также связанные с ней налоги и отчисления в государственный бюджет.

Основная заработная плата исполнителей на конкретное ПО рассчитывается по формуле:

Зоi = ∑ni=1 \* Зci \* Фрi \* К (5.6 )

где n− количество исполнителей, занятых разработкой конкретного ПО;

−среднедневная заработная плата i-го исполнителя (*руб.*);

− плановый фонд рабочего времени i-го исполнителя (дн.);

К − коэффициент премирования.

Основная заработная плата исполнителей на конкретное ПО рассчитывается по формуле:

|  |  |
| --- | --- |
|  | (5.7) |

где n – количество исполнителей;

– дневная тарифная ставка i-го исполнителя, руб.;

– эффективный фонд рабочего времени i-го исполнителя, дн.;

– коэффициент премий.

Учитывая, что коэффициент премирования в данном случае равен 1,8, основная заработная плата будет равна:

**5.2.2.2 Расчет дополнительной заработной платы**

Дополнительная заработная плата на конкретный программный продукт определяется по нормативу в процентах к основной заработной плате:

 , (5.8)

где Здi− дополнительная заработная плата исполнителей на конкретное   
ПО (д.e.).

Нд− норматив дополнительной заработной платы рассчитывается по  
формуле

Приняв во внимание, что размер норматива дополнительной заработной платы по данному проекту равен 25%, дополнительная заработная плата примет следующий вид:

.

**5.2.2.3 Расчет отчислений в фонд социальной защиты населения**

Отчисления в фонд социальной защиты населения производятся в момент начисления заработной платы. Их размер (в процентном выражении) устанавливает законодательно. Сумма отчислений на социальные нужды зависит от основной и дополнительной заработной платы и может быть рассчитана по следующей формуле:

, (5.9)

где Нсз− норматив отчислений в фонд социальной защиты населения (%).

**5.2.2.4 Расчет отчислений по обязательному страхованию**

Отчисления по обязательному страхованию определяется по нормативу, его аналогии с фондом социальной защиты населения:

 , (5.10)

где Hос− норматив отчислений по обязательному страхованию (%).

.

**5.2.2.5 Расчет расходов по статье «Материалы»**

Расходы по статье «Материалы» (М) определяются на основании сметы затрат, разрабатываемой на ПО с учетом действующих нормативов. По статье «Материалы» отражаются расходы на магнитные носители, бумагу, красящие ленты и другие материалы, необходимые для разработки ПО. Нормы расхода материалов в суммарном выражении (Hм) определяются в расчете на 100 строк исходного кода или по нормативу в процентах к фонду основной заработной платы разработчиков (Нмз), который устанавливается организацией (3 – 5 %). В процессе разработки данного программного продукта, принимается норма расхода материалов в размере 5 % от основной заработной платы. Сумма затрат на расходные материалы рассчитывается по формуле:

, (6.11)

где Нмз− норма расхода материалов от основной заработной платы (%).

.

**5.2.2.6 Расчет расходов по статье «Спецоборудование»**

Данная статья включается в смету расходов на разработку ПО в том случае, когда приобретаются специальное оборудование или специальные программы, предназначенные для разработки и создания только данного ПО:

, (5.12)

где Цсi− стоимость конкретного специального оборудования (д.е.);

n− количество применяемого специального оборудования.

Для введения в эксплуатацию данного программного средства необходимо приобретение сервера и доменного имени.

Таким образом, приобретены и использованы:

сервер – 90,00 рублей;

доменное имя – 40 рублей.

Рсi = 90 + 40 =130 (руб).

**5.2.2.7 Расчет расходов по статье «Машинное время»**

Расходы по статье «Машинное время» (Рмi) включают оплату машинного времени, необходимого для разработки и отладки ПО, которое определяется по нормативам (в машино-часах) на 100 строк исходного кода (Hмв)   
машинного времени в зависимости от характера решаемых задач и типа ПК   
(прил. 6):

, (5.13)

где Цмi− цена одного машино-часа (д.е.);

Voi− общий объем ПО (строк исходного кода);

Нмв− норматив расхода машинного времени на отладку 100 строк исходного кода (машино-часов).

За рыночною стоимость одного машино-часа компьютера со всем необходимым оборудованием примем 0,9 рублей.

Рмi = 0,9 \*35,4\*12 = 382,23 (руб).

**5.2.2.8 Расчет расходов по статье «Прочие затраты»**

Расходы по статье «Прочие затраты» (Пзi) на конкретное ПО включают затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы. Определяются по нормативу, разрабатываемому в целом по организации, в процентах к основной заработной плате:

, (5.14)

где Hпз− норматив прочих затрат в целом по организации.

Пзi = 362,34 \* 0,2 = 72,47 (руб).

**5.2.2.9 Расчет затрат по статье «Накладные расходы»**

Затраты по содержанию административно-управленческого аппарата, различного рода общехозяйственные затраты отражаются по статье «Накладные расходы» и относятся на конкретный программный продукт по нормативу в процентном отношении к основной заработной плате. Норматив отнесения расходов по статье «Накладные расходы» на данный программный продукт в данной организации равен 20 % от основной заработной платы.

Норматив устанавливается в целом по организации:

, (5.15)

где Pнi– накладные расходы на конкретную ПО (д.е.);

Нрн – норматив накладных расходов в целом по организации.

Норматив накладных расходов определяется по формуле:

, (5.16)

где Рн– накладные расходы в целом по организации (д.е.).

Рнi = 362,34 \* 0,2 = 72,47 (руб).

**5.2.2.10 Расчет общей суммы расходов по смете**

Общая сумма расходов по смете (Спi) на ПО рассчитывается по формуле:

Спi = Зoi + Здi + Зcзi + Зосi+ Мi + Pci + Pмi + Пзi + Pнi. (5.17)

Cпi = 1885,95 + 471,49 + 801,53 + 141,45 + 18,12 +130 + 382,23 + 72,47 + 72,47 = 3975,71 (руб).

**5.2.2.11 Определение цены и оценка эффективности ПО**

Прибыль рассчитывается по формуле:

, (5.18)

где Поi− прибыль от реализации ПО заказчику (д.е.);  
Урпi− уровень рентабельности ПО (%);

Спi− себестоимость ПО (д.е.).

Расчитаем прибыль от реализации ПО заказчику или на рынке, с учетом ожидаемого уровня рентабельности в 40 %.

Псi = 3975,71 \* 0,4 = 1590,28 (руб).

Прогнозируемая цена ПО без налогов (Цпi):

Цпi= Cpi+Пci (5.19)

Цпi = 3975,71+1590,28 = 5565,99 (руб).

Налог на добавленную стоимость (НДСi):

НДСi = ЦПI \* НДС : 100 (5.20)

где Hдc− норматив НДС (%). На сегодняшний день, ставка НДС составляет 20%.

НДСi = 5565,99\*0,2 = 1113,20 (руб).

Прогнозируемая отпускная цена (Цoi):

Цoi = Цпi+ НДСi (5.21)

Цоi = 5565,99+1113,20 = 6679,19 (руб).

В зависимости от договорных условий компания-разработчик, в основном, покрывает затраты на разработку программного продукта, в том числе и прибыль, за счет средств заказчика.

В настоящее время в Республике Беларусь действует налог на прибыль. Его ставка устанавливается законодательно (на данный момент его размер составляет 18 %).

Чистая прибыль организации-разработчика от реализации программного продукта определяется путем вычитания налога на прибыль:

|  |  |
| --- | --- |
|  | (5.22) |

где – ставка налога на прибыль.

Таким образом, проектируемый программный продукт является эффективным для организации-разработчика, сумма чистой прибыли составит руб.

В разделе были произведены экономические расчеты, позволяющие определить затраты на разработку программного средства, а также определить прогнозируемую отпускную цену всего проекта.

Таблица 5.3 содержит результаты расчетов экономических показателей, проведенных в данном разделе.

Проанализировав полученные результаты можно сделать вывод, что в результате реализации данного ПО пользователи могут понести внушительные капитальные затраты на его приобретение, чему свидетельствует высокая цена ПО: с учетом НДС она составляет 6679,19 рублей. С другой стороны, данное ПО существенно облегчает работу пользователей, что способствует увеличению прибыли клиентов, а значит, и скорой окупаемости.

Таблица 5.3 – Расчет затрат на создание ПО и отпускной цены

|  |  |
| --- | --- |
| Наименование показателя | Сумма, руб. |
| Основная заработная плата |  |
| Дополнительная заработная плата |  |
| Отчисления в Фонд социальной защиты населения | 801,53 |
| Отчисления по обязательному страхованию | 141,45 |
| Расходы по статье «Материалы» | 18,12 |
| Расходы по статье «Спецоборудование» | 130,00 |
| Расходы по статье «Машинное время» | 382,23 |
| Расходы по статье «Прочие затраты» | 72,47 |
| Расходы по статье «Накладные расходы» | 922,6 |
| Полная себестоимость ПО | 3975,71 |
| Чистая прибыль от реализации ПО | 1304,03 |
| Планируемая цена разработчика ПС без налогов | 5565,99 |
| Налог на добавленную стоимость | 1113,20 |
| Отпускная цена с НДС | 6679,19 |

При реализации данного ПО будет получена чистая прибыль в размере 1304,03 рублей.

Выбор проекта связан с расчетом экономического эффекта, который может определятся как у разработчика, так и у пользователя.

Для разработчика экономический эффект выступает в виде чистой прибыли от реализации, остающейся в распоряжении организации, а у пользователя — в виде экономии трудовых, материальных и финансовых ресурсов, получаемый за счет:

* снижения трудоемкости расчетов и алгоритмизации программирования и отладки программ за счет использования ПС в процессе разработки;
* сокращение расходов на оплату машинного времени и других ресурсов на отладку программ;
* снижение расходов на материалы;
* улучшение показателей основной деятельности в результате использования ПО.

Стоимостная оценка ПС у разработчиков предполагает определение затрат, что включает следующие статьи:

* заработная плата исполнителей — основная и дополнительная;
* отчисления в Фонд социальной защиты населения и по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний;
* расходы на материалы и комплектующие;
* расходы на спецоборудование;
* расходы на оплату машинного времени.

**5.2.2.12 Оценка эффективности использования ПО у заказчика**

Для оценки эффективности использования ПО у заказчика нам необходимо провести оценку всех понесенных заказчиком затрат, а также оценить период их окупаемости и возможность получения прибыли.

Учитывая особенности данной программы, для дальнейшего получения прибыли полгода сайт будет работать бесплатно, с целью привлечения клиентов и продвижения сайта. При этом в заказчик понесет следующие затраты, которые отображены в таблице 5.3.

Таблица 5.3 – Расчет экономического эффекта от использования нового программного средства

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Показатели | Единицы  измерения | Годы | | | |
| 2017 | 2018 | 2019 | 2020 |
| 1 | 2 | 3 | 4 | 5 | 6 |
| Прогнозируемая прибыль | руб. | 6000,00 | 12000,00 | 12000,00 | 12000,00 |
| Затраты |  |  |  |  |  |
| Приобретение ПО (Кпр) | руб. | 6679,90 |  |  |  |
| Оплата труда ИТ-специалиста (удаленная работа на 0,5 ставки) | руб. | 4000,00 | 4000,00 | 4000,00 | 4000,00 |
| Отчисления в фонд социальной защиты населения | руб. | 1360,00 | 1360,00 | 1360,00 | 1360,00 |
| Отчисления по обязательному страхованию | руб. | 240,00 | 240,00 | 240,00 | 240,00 |
| Приобретение домена | руб. | 300,00 | 300,00 | 300,00 | 300,00 |
| Приобретение сервера | руб. | 400,00 | 400,00 | 400,00 | 400,00 |
| Приобретение и размещение курсов | руб. | 1000,00 | 1000,00 | 1000,00 | 1000,00 |
| Всего затрат | тыс. руб. | 13979,9 | 7300,00 | 7300,00 | 7300,00 |
| То же с учетом фактора времени | тыс. руб. | 13979,9 | 6348,08 | 5519,53 | 4799,75 |
| Экономический эффект |  |  |  |  |  |
| Превышение результата над затратами | тыс. руб. | - 7979,9 | 5651,92 | 6480,47 | 7200,25 |
| То же с нарастающим итогом | тыс. руб. | - 7979,9 | -2327,98 | 4152,49 | 11352,74 |
| Коэффициент приведения | единицы | 1 | 0,8696 | 0,7561 | 0,6575 |

Рассчитаем прогнозируемую прибыль заказчика. Предусмотрено, что первые полгода заказчик не будет получать прибыль от использования ПО, это время будет потрачено на продвижение сайта. По опыту реализации подобных проектов к этому времени должно зарегистрироваться на сайте, для прохождения онлайн-обучения, приблизительно 2,5 тыс. человек. По истечению полугода для прохождения онлайн-обучения на данном сайте каждому пользователю необходимо будет приобрести платный аккаунт. Из опыта реализации подобных проектов, соглашается, как правило, 40 % от общего количества пользователей. В нашем случае это 1000 человек. За вторую половину года на сайте получат платный аккаунт еще около 500 человек. Стоимость аккаунта составляет 8,00 рублей. В год прибыль будет составлять: 1500 \* 8,00=12000,00 рублей.

Таким образом, все затраты заказчика окупятся на 3 году эксплуатации ПО. Проект представляется эффективным и полезным для заказчика.

ЗАКЛЮЧЕНИЕ

В ходе работы над дипломным проектом проанализированы системы онлайн обучения с использованием информационных технологий. Исследованы различные направления и подходы к решению подобных задач.

Проведен анализ предметной области, исследованы существующие аналоги. Результатом этого анализа явилось обобщение достоинств и недостатков существующих решений, которые учтены при разработке функциональных требований к разработанному программному средству. Наиболее часто встречающимся недостатком у имеющихся решений оказалось отсутствие комплексного решения всех поставленных требований.

На основе функциональных требований было произведено проектирование программного средства. В нем представлены разработка архитектуры ПС, разработка модели базы данных, разработка back-end и front-end частей веб-приложения. В разделе разработки архитектуры ПС приведена обобщенная схема работы программного средства. Также детально рассмотрена структура и используемые методы и приемы при разработке Ruby on Rails веб-приложений. В разделе представлены диаграммы классов моделей и классов контроллеров. Так же рассмотрено модульное тестирование.

Согласно требованиям были сформированы тестовые наборы, которые успешно пройдены в ходе тестовых испытаний веб-приложения. Успешность прохождения тестов показывает корректность работы программного средства, соответствие функциональным требованиям.

На завершающем этапе подробно описаны возможности и методика использования программного средства, которая позволяет за достаточно быстрые сроки освоить работу с программой.

Также в ходе работы над дипломным проектом рассмотрена экономическая сторона проектирования и разработки программного средства, а также рассчитан экономический эффект от внедрения программного средства и показатели эффективности использования программного средства как для пользователя, так и для заказчика. В результате расчётов подтвердилась целесообразность разработки.

Главной целью при разработке программного средства было поставлено устранение основных недостатков существующих аналогов, а также разработка и реализация всего основного функционала. В ходе работы над дипломным проектом эта цель была успешно достигнута: разработано веб-приложение для оналйн обучения с поддержкой мультимедийного контента.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Воробьева Т.А. К вопросу о понятии электронного обучения / Т.А. Воробьева // Идеи и идеалы. - № 1 (19). – 2014. – С. 143-152.

[2] Online – Defi nition of Online by Macmillan Dictionary [Электронный ресурс]. – Режим доступа: http://www.macmillandictionary. com/dictionary/british/online.

[3] Online – Defi nition and more from the free Merriam-Webster dictionary [Электронный ресурс]. – Режим доступа: <http://www.merriam-webster.com/dictionary/online>.

[4] Online | Defi ne Online at Dictionary.com. [Электронный ресурс]. – Режим доступа: http://dictionary.reference.com/browse/online?s= b&path=/.

[5] Казанская О.В. От дистанционного обучения к электронному. 2009. Информ. Технологии в образовании. – Новосибирск : Изд-во НГТУ, 2009. – № 1 (17). – С. 4–5.

[6] Кузьмин В. Что это такое - дистанционное обучение?// В. Кузьмин/ [Электронный русурс]. – Режим доступа: [https://shkolazhizni.ru/school/articles/ 38778/](https://shkolazhizni.ru/school/articles/%2038778/).

[7] Википедия [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Coursera>.

[8] Курсы «ИнтерКОЛО» [Электронный ресурс]. – Режим доступа: http://www.interkolo.com/page-job\_free\_education.html.

[9] Обучение онлайн [Электронный ресурс]. – Режим доступа: http://ru.osvita.ua/abroad/higher\_school/distance-learning/37601/.

[10] THEORY&PRACTICE [Электронный ресурс]. – Режим доступа: https://theoryandpractice.ru/posts/8484-onlayn-obrazovanie-na-russkom.

[11] Обучения и мероприятия [Электронный ресурс]. – Режим доступа: <http://www.cisco.com/c/ru_ru/training-events.html>.

[12] Почему Ruby / Ruby on Rails? [Электронный ресурс]. – Режим доступа: <https://medium.com/evrone-ru>.

[13] Архитектура REST [Электронный ресурс]. – Режим доступа: <https://habrahabr.ru/post/38730/>.

[14] Что такое модульное тестирование? Значение термина модульное тестирование [Электронный ресурс]. – Режим доступа: <http://animatika.ru/info/gloss/unit-testing.html>.

[15] Функциональное тестирование программного обеспечения [Электронный ресурс]. – Режим доступа: <http://aplana.ru/services/testing/> functionalnoe-testirovanie.

[16] Online Etymology Dictionary URL: http:// www.etymonline.com/index.php?term=online& allowed\_in\_frame=0/.

ПРИЛОЖЕНИЕ А

**(справочное)**

**Исходный код веб-приложения**

class *ApplicationController* < *ActionController*::*Base*

*include DeviseTokenAuth*::*Concerns*::*SetUserByToken*

*include DeviseTokenAuth*::*Concerns*::*SetUserByToken*

*include Pundit*

*protect\_from\_forgery* with: :null\_session

rescue\_from *ActionController*::*ParameterMissing*, with: :invalid\_params

rescue\_from *ActiveRecord*::*RecordNotFound*, with: :not\_found

rescue\_from *Pundit*::*NotAuthorizedError*, with: :not\_authorized

def *render\_error\_messages*(*object*, *status* = :unprocessable\_entity)

*render* json: { errors: *object*.errors.messages, fields: *object*.errors.keys },

status: *status*, head: *status*

end

def *render\_success*(*status* = true)

*render* json: { success: *status* }, status: :ok

end

def *invalid\_params*(*exception*)

*render* json: { errors: "Required parameter is missing: #{*exception*.param}" }, status: :unprocessable\_entity, head: :unprocessable\_entity

end

def *not\_found*

*render* json: { errors: "Item not found" }, status: :not\_found, head: :not\_found

end

def *not\_authorized*

*render* json: { errors: "Not allowed" }, status: 403, head: 403

end

*private*

def *current\_user*

@current\_user ||= *Admin*.*find*(session[:user\_id]) if session[:user\_id]

end

*helper\_method* :current\_user

def *authorize*

*redirect\_to* login\_url, alert: "Not authorized" if current\_user.nil?

end

end

class *CommentsController* < *ApplicationController*

*before\_action* :set\_video, except: [ :destroy ]

*before\_action* :authenticate\_user!

def *index*

*render* json: @video.comments

end

def *create*

*comment* = @video.comments.new(comment\_params)

if *comment*.save

*render* json: *comment*

else

render\_error\_messages(*comment*, 400)

end

end

def *destroy*

*comment* = *Comment*.*find*(params[:id])

if *comment*.delete

*render* json: *comment*

else

render\_error\_messages(*comment*, 400)

end

end

*private*

def *comment\_params*

{

body: params[:body],

commentable\_id: params[:item\_id],

commentable\_type: params[:type],

user\_id: params[:sender\_id],

root\_comment\_id: params[:root\_comment\_id]

}

end

def *set\_video*

@video = *Video*.*find*(params[:item\_id])

end

end

class *ConversationsController* < *ApplicationController*

*before\_action* :authenticate\_user!

*before\_action* :get\_mailbox

*before\_action* :get\_conversation, except: [:index, :empty\_trash]

*before\_action* :get\_box, only: [:index]

def *index*

if @box.eql? "inbox"

@conversations = @mailbox.inbox

elsif @box.eql? "sent"

@conversations = @mailbox.sentbox

else

@conversations = @mailbox.trash

end

@conversations = @conversations.*paginate*(page: params[:page], per\_page: 10)

*render* json: @conversations, each\_serializer: *ConversationSerializer*

end

def *show*

end

def *mark\_as\_read*

@conversation.mark\_as\_read(current\_user)

end

def *reply*

current\_user.reply\_to\_conversation(@conversation, params[:body])

end

def *destroy*

@conversation.move\_to\_trash(current\_user)

end

def *restore*

@conversation.untrash(current\_user)

end

def *empty\_trash*

@mailbox.trash.each do |*conversation*|

*conversation*.receipts\_for(current\_user).update\_all(deleted: true)

end

end

*private*

def *get\_mailbox*

@mailbox ||= current\_user.mailbox

end

def *get\_conversation*

@conversation ||= @mailbox.conversations.*find*(params[:id])

end

def *get\_box*

if params[:box].blank? or !["inbox","sent","trash"].include?(params[:box])

params[:box] = 'inbox'

end

@box = params[:box]

end

end

class *CoursesController* < *ApplicationController*

*before\_action* :authenticate\_user!, except: [ :index ]

*before\_action* :set\_course, only: [:destroy]

*skip\_before\_filter* :verify\_authenticity\_token

def *index*

if params[:all]

@courses = *Course*.all

else

authorize :courses

@courses = policy\_scope(:Courses)

end

*render* json: @courses, each\_serializer: *CourseSerializer*

end

def *show*

end

def *destroy*

@course.delete ? (*render* json: ({id: params[:id]}.to\_json)) :

render\_error\_messages(@course, 400)

end

def *create*

authorize :courses

*course\_form* = *CourseForm*.new(params)

if *course\_form*.save(current\_user.id)

render\_success

else

render\_error\_messages(course, 400)

end

end

*private*

def *set\_course*

@course = *Course*.*find*(params[:id])

end

end

class *MessagesController* < *ApplicationController*

*before\_action* :authenticate\_user!

def *new*

@chosen\_recipient = *User*.find\_by(id: params[:to].to\_i) if params[:to]

end

def *create*

*recipients* = *User*.*where*(id: params['recipients'])

*conversation* = current\_user.send\_message(*recipients*, params[:message][:body], params[:message][:subject]).conversation

if *conversation*

*render* json: *conversation*

else

render\_error\_messages(*conversation*, 400)

end

end

end

class *SessionsController* < *ApplicationController*

def *new*

end

def *create*

*user* = *Admin*.find\_by(email: params[:session][:email])

if *user* && *user*.test\_password(params[:session][:password], *user*.password)

session.clear

session[:user\_id] = *user*.id

*redirect\_to* admin\_url

else

flash[:alert] = "Email or password is invalid"

*render* "new"

end

end

def *destroy*

session[:user\_id] = nil

*redirect\_to* login\_url

end

end

class *SubscriptionsController* < *ApplicationController*

*before\_action* :authenticate\_user!

*before\_action* :set\_subscription\_by\_user, only: [:destroy]

def *add\_course*

unless current\_user.subscription.courses.find\_by(id: course\_id)

current\_user.subscription.courses << *Course*.find\_by(id: course\_id)

render\_success

else

*render* json: {error: 'course has already been created'}, status: 400

end

end

def *destroy*

*course* = @subscription.courses.*find*(params[:id])

if @subscription.courses.delete(*course*)

*render* json: {id: params[:course\_id]}.to\_json

else

render\_error\_messages(@subscription, 400)

end

end

def *check\_status*

if current\_user.subscription.courses.find\_by(id: course\_id)

render\_success

else

render\_success(false)

end

end

*private*

def *course\_id*

params.*require*(:id)

end

def *set\_subscription\_by\_user*

@subscription = *Subscription*.*find*(current\_user.id)

end

end

class *TasksController* < *ApplicationController*

*before\_action* :authenticate\_user!

*before\_action* :set\_task, only: [:destroy]

def *destroy*

if @task.delete

*render* json: {id: params[:id]}.to\_json

else

render\_error\_messages(@task, 400)

end

end

def *create*

*tasks* = *Task*.create\_tasks(params[:id].to\_i, tasks\_params)

*render* json: {tasks: *tasks*}.to\_json

end

*private*

def *tasks\_params*

params[:tasks].values

end

def *set\_task*

@task = *Task*.*find*(params[:id])

end

end

class *Course* < *ActiveRecord*::*Base*

*belongs\_to* :user

*has\_many* :videos, dependent: :destroy

*has\_many* :tasks, dependent: :destroy

*has\_and\_belongs\_to\_many* :subscriptions

*has\_and\_belongs\_to\_many* :categories

*accepts\_nested\_attributes\_for* :categories, allow\_destroy: true

*has\_one* :image, dependent: :destroy

*validates\_uniqueness\_of* :name

def *video\_meta*

{ name: name, description: description, course\_id: id,

image: image.full\_url(:course\_sketch), user\_id: user\_id,

tasks: tasks, domain: *Settings*.domain\_name }

end

end

class *Image* < *ActiveRecord*::*Base*

mount\_uploader :file, *PhotoUploader*

def *full\_url*(*format*)

*Settings*.domain\_name + file.url(*format*.to\_sym)

end

end

class *Task* < *ActiveRecord*::*Base*

*belongs\_to* :course

mount\_uploader :file, *TaskUploader*

def self.*create\_tasks*(*id*, *items*)

*course* = *Course*.*find*(*id*)

*tasks* = []

*items*.each do |*task*|

*tasks* << *Task*.create(name: *task*.original\_filename.rpartition('.').first, file: *task*.tempfile, format:

*task*.original\_filename.rpartition('.').last)

end

*course*.tasks.push(\**tasks*)

*tasks*

end

def self.*parsing\_to\_array*(*items*)

*tasks* = []

*items*.each do |*item\_name*|

*tasks* << *Task*.find\_by(name: *item\_name*)

end

*tasks*

end

end

class *User* < *ActiveRecord*::*Base*

rolify

devise :database\_authenticatable, :registerable,

:recoverable, :rememberable, :trackable, :validatable,

:confirmable, :omniauthable

*include DeviseTokenAuth*::*Concerns*::*User*

*has\_many* :votes

*has\_many* :courses

*has\_many* :tasks, through: :courses

*has\_many* :comments

*belongs\_to* :role

*has\_one* :subscription

*has\_and\_belongs\_to\_many* :tasks

acts\_as\_messageable

*before\_create* :default\_users\_role, :add\_subscription

def *vote\_init*(*vote\_type*, *item\_id*, *item\_type*)

case *vote\_type*

when 'like'

*like* = voted\_up\_on(*item\_id*,*item\_type*)

*like* ? *like*.delete : like(*item\_type*,*item\_id*)

when 'dislike'

*dislike* = voted\_down\_on(*item\_id*,*item\_type*)

*dislike* ? *dislike*.delete : dislike(*item\_type*,*item\_id*)

end

end

def *voted\_up\_on*(*item\_id*, *item\_type*)

*Vote*.find\_by(user\_id: self.id, votable\_id: *item\_id*,

votable\_type: *item\_type*.downcase, vote\_flag: true)

end

def *voted\_down\_on*(*item\_id*, *item\_type*)

*Vote*.find\_by(user\_id: self.id, votable\_id: *item\_id*,

votable\_type: *item\_type*.downcase, vote\_flag: false)

end

def *like*(*type*,*id*)

*vote* = *Vote*.create(votable\_id: *id*, votable\_type: *type*.downcase, vote\_flag: true)

self.votes << *vote*

end

def *dislike*(*type*,*id*)

*vote* = *Vote*.create(votable\_id: *id*, votable\_type: *type*.downcase, vote\_flag: false)

self.votes << *vote*

end

def *default\_users\_role*

add\_role :user

end

def *add\_subscription*

self.subscription = *Subscription*.create!

end

def *mailboxer\_email*(*object*)

email

end

end

class *Video* < *ActiveRecord*::*Base*

*belongs\_to* :course

*has\_one* :image, dependent: :destroy

*has\_many* :votes, as: :votable, dependent: :destroy

*has\_many* :comments, as: :commentable, dependent: :destroy

mount\_uploader :video, *VideoUploader*

*validates\_uniqueness\_of* :name

def *full\_url*

*Settings*.domain\_name + video.url

end

end

class *Vote* < *ActiveRecord*::*Base*

*belongs\_to* :votable, polymorphic: true

*belongs\_to* :user

def self.*get\_dislikes*(*type*, *id*)

*Vote*.*where*(vote\_flag: false, votable\_type: *type*, votable\_id: *id*)

end

def self.*get\_likes*(*type*, *id*)

*Vote*.*where*(vote\_flag: true, votable\_type: *type*, votable\_id: *id*)

end

end

.module('Video')

.controller('CurrentVideoCtrl', CurrentVideoCtrl);

CurrentVideoCtrl.$inject = [

'ENV', '$scope', '$state', 'Video', '$stateParams', '$sce', 'Vote'

];

function CurrentVideoCtrl(ENV, $scope, $state, Video, $stateParams, $sce, Vote) {

console.log('CurrentVideoCtrl load');

var vm = this;

vm.video = undefined;

vm.config = undefined;

vm.vote = vote;

vm.checkVoter = checkVoter;

getVideo();

function getVideo() {

Video.get({id: $stateParams.id}).$promise

.then(function(data) {

vm.video = data.video;

vm.config = {

sources: [ {

src: $sce.trustAsResourceUrl(data.video.url),

type: data.video.format

} ]

};

})

.catch(function() {

console.log('courses load error');

});

}

function vote(vote\_type) {

var status = JSON.parse(window.localStorage['status'] || '{}');

Vote.charge({id: $stateParams.id, user\_email: status.name, vote\_type: vote\_type, item\_type: 'video'}).$promise

.then(function(data) {

vm.video = data.vote\_video;

})

.catch(function() {

console.log('courses load error');

});

}

function checkVoter(author\_id) {

if (JSON.parse(window.localStorage['status']).id == author\_id)

return true;

else

return false

}

.*module*('Video')

.factory('Vote', ['$resource' ,'ENV', function($resource, ENV) {

return $resource(ENV.apiEndpoint + "/users/vote",

{

id: '@id'

},{

charge: {method:'POST', params:{id: '@id'}}

});

}])

.directive('vote', function(Vote) {

return {

restrict: 'E',

scope: {

video: '='

},

templateUrl: 'components/video/vote.html',

link: function (scope, element, attrs) {

scope.vote = *vote*;

scope.checkSender = *checkSender*;

function *vote*(vote\_type) {

var status = *JSON*.parse(*window*.localStorage['status'] || '{}');

Vote.charge({id: $stateParams.id, user\_email: status.name, vote\_type: vote\_type, item\_type: 'video'}).$promise

.then(function(data) {

scope.video = data.vote\_video;

})

.catch(function() {

console.log('courses load error');

});

}

function *checkSender*(sender\_comment\_id) {

if (*JSON*.parse(*window*.localStorage['status']).id == sender\_comment\_id)

return false;

else

return true

}

}

};

});

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Обозначение* | | | | | *Наименование* | *Дополнительные сведения* | | | | |
|  | | | | | Текстовые документы |  | | | | |
|  | | | | |  |  | | | | |
| БГУИР ДП 1–40 01 03 00 066 ПЗ | | | | | Пояснительная записка | 85 c | | | | |
|  | | | | | Отзыв руководителя |  | | | | |
|  | | | | | Рецензия |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | | Графические документы |  | | | | |
|  | | | | |  |  | | | | |
| ГУИР.400103.066 ПД1 | | | | | Диаграмма деятельности. | Формат А1 | | | | |
|  | | | | | Чертеж. |  | | | | |
|  | | | | |  |  | | | | |
| ГУИР.400103.066 ПД2 | | | | | Диаграмма классов | Формат А1 | | | | |
|  | | | | | (моделей). Чертеж. |  | | | | |
|  | | | | |  |  | | | | |
| ГУИР.400103.066 ПД3 | | | | | Диаграмма классов | Формат А1 | | | | |
|  | | | | | (контроллеров). Чертеж. |  | | | | |
|  | | | | |  |  | | | | |
| ГУИР.400103.066 ПД4 | | | | | Диаграмма front-end | Формат А1 | | | | |
|  | | | | | модулей. Чертеж. |  | | | | |
|  | | | | |  |  | | | | |
| ГУИР.400103.066 ПЛ1 | | | | | Диаграмма вариантов | Формат А1 | | | | |
|  | | | | | использования. Плакат. |  | | | | |
|  | | | | |  |  | | | | |
| ГУИР.400103.066 ПЛ2 | | | | | Схема базы данных. | Формат А1 | | | | |
|  | | | | | Плакат. |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  |  |  |  |  | *БГУИР ДП 1- 40 01 03 00 066 Д1* | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| *Изм* | *Л.* | *№ докум.* | *Подп.* | *Дата* | ***Веб-приложение для онлайн обучения с поддержкой мультимедийного контента.***  ***Ведомость дипломного  проекта*** |  | | | *Лист* | *Листов* |
| *Разраб.* | | ***Михаськов К.А.*** |  |  | *И* |  |  | *85* | *85* |
| *Пров.* | | ***Жвакина А.В.*** |  |  | ***Информатика***  ***гр. 253503*** | | | | |
| *Т.контр.* | | ***Жвакина А.В.*** |  |  |
| *Н.контр.* | | ***Шиманский В.В.*** |  |  |
| *Утв.* | | ***Волорова Н.А.*** |  |  |