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

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

09.03.03 ПРИКЛАДНАЯ ИНФОРМАТИКА

ПРИКЛАДНАЯ ИНФОРМАТИКА В СОЦИАЛЬНОЙ СФЕРЕ

БАКАЛАВРСКАЯ РАБОТА

на тему: **Обучающая информационная система по курсу «Разработка программных приложений»**

Студент В.А. Качанов _____
(личная подпись)

Руководитель Е.А Ерофеева _____
(личная подпись)

Допустить к защите
Заведующий кафедрой к.тех.н, доцент, А.В. Очеповский _____

«_____»_____20_____г.

Тольятти 2016

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

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

УТВЕРЖДАЮ
Зав. кафедрой «Прикладная математика и информатика»
_____А.В.Очеповский

«____»_____20__г.

ЗАДАНИЕ
на выполнение бакалаврской работы

Студент _____Качанов Виктор Андреевич_____

1. Тема: Обучающая информационная система по курсу “Разработка программных приложений”

2. Срок сдачи студентом законченной выпускной квалификационной работы: 10.06.2016

3. Исходные данные к выпускной квалификационной работе: режим функционирования: круглосуточный (365/24/7); количество одновременно работающих пользователей: не менее 3; требования к технологиям реализации: в качестве СУБД необходимо использовать MySQL._____

4. Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов, разделов):

анализ учебной и учебно-методической литературы по проблеме построения систем обучения, а также анализ IT аналогов; постановка задачи реализации и внедрения информационной системы; обоснование выбора технологии проектирования информационной системы; моделирование информационной системы; построение функциональной и структурной схемы реализации автоматизированной системы; обоснование выбора средств реализации информационной

системы; реализация системы обучения для организации дистанционного обучения на примере курса “Разработка программных приложений”.

5. Ориентировочный перечень графического и иллюстративного материала: диаграммы функционального моделирования IDEF0; диаграммы потоков данных DFD; диаграммы объектного моделирования UML; экранные формы, демонстрирующие работоспособность разработанной информационной системы; презентация.

6. Дата выдачи задания « 11 » января 2016 г.

Заказчик начальник отдела разработки информационных систем ЦНИТ ТГУ, к. ф. – м. н.	(подпись)	С.В. Баумгертнер
Руководитель выпускной квалификационной работы	(подпись)	Е.А. Ерофеева
Задание принял к исполнению	(подпись)	В. А. Качанов

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

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

УТВЕРЖДАЮ
Зав. кафедрой «Прикладная математика и информатика»
_____А.В.Очеповский

«____»_____2016 г.

КАЛЕНДАРНЫЙ ПЛАН
выполнения бакалаврской работы

Студента _____Качанова Виктора Андреевича_____
по теме: Обучающая информационная система по курсу “Разработка программных приложений”

Наименование раздела работы	Плановый срок выполнения раздела	Фактический срок выполнения раздела	Отметка о выполнении	Подпись руководителя
Выбор и утверждение темы ВКР	13.01.2016	13.01.2016	Выполнено	
Поиск и анализ литературы по проблеме организации дистанционных курсов	18.01.2016	18.01.2016	Выполнено	
Анализ бизнес-процессов организации курса “Разработка программных приложений”	25.01.2016	25.01.2016	Выполнено	
Концептуальное моделирование предметной области	15.02.2016	15.02.2016	Выполнено	
Обоснование выбора средств реализации информационной системы	29.02.2016	29.02.2016	Выполнено	
Проектирование БД	7.03.2016	7.03.2016	Выполнено	
Проектирование интерфейса обучающей	18.03.2016	18.03.2016	Выполнено	

системы				
Разработка функциональной и организационной структуры ИС	23.03.2016	23.03.2016	Выполнено	
Реализация обучающей системы	1.04.2016	1.04.2016	Выполнено	
Тестирование и корректировка программного приложения	22.04.2016	22.04.2016	Выполнено	
Оформление пояснительной записки ВКР	09.05.2016	09.05.2016	Выполнено	
Разработка презентации для защиты	23.05.2016	23.05.2016	Выполнено	
Предзащита выпускной квалификационной работы	06.05.2016	06.05.2016	Выполнено	
Корректировка ВКР согласно сделанным замечаниям	09.05.2016	09.05.2016	Выполнено	
Проверка выпускной квалификационной работы в системе «Антиплагиат ВУЗ»	10.06.2016	10.06.2016	Выполнено	
Оформление документов к защите ВКР	13.06.2016	13.06.2016	Выполнено	
Сдача оформленной пояснительной записки ВКР и разработанного программного приложения	17.06.2016	17.06.2016	Выполнено	

Руководитель выпускной квалификационной работы _____ Е.А Ерофеева
(подпись)

Задание принял к исполнению _____ В.А. Качанов
(подпись)

Аннотация

Тема: «Обучающая информационная система по курсу “Разработка программных приложений»

Актуальность выбора темы работы обусловлена решением **проблемы**, связанной с недостаточной наглядностью учебного процесса по дисциплине “Разработка программных приложений”. В процессе анализа существующих аналогов был сделан вывод о том, что на текущий момент существует множество различных инструментов для обучения разработке прикладных программ, но не один не обладает всеобъемлющим спектром практических заданий, которые могли бы в простой и в то же время понятной форме изложить учебный материал.

Объем бакалаврской работы 52 страницы. В работе: рисунков - 27, таблиц - 9, список литературы состоит из 25 источников.

Целью работы является практическая реализация обучающей системы для повышения эффективности учебного процесса.

Бакалаврская работа состоит из введения, трех глав, заключения и приложений. Во введении обосновывается актуальность выбранной темы, формируется проблема, определяются объект и предмет исследования, выделяются цели и задачи.

В первой главе происходит анализ предметной области, проводится сравнительный анализ современных методологий и технологий концептуального моделирования информационных систем.

Во второй главе описывается проектирование модулей информационной системы

В третьей главе описывается программная реализация модулей информационной системы.

В заключении сделаны основные выводы и итоги по проделанной работе.

Оглавление

Введение

В современном информационном обществе наличие большого объема информации привело к практически повсеместному использованию информационных технологий во всех учебных заведениях, потому что от того насколько эффективнее будет организован учебный процесс будет зависеть качество образования, а значит культурное и социально-экономическое развитие общества.

Организация учебного процесса представляет собой набор методических мер по проведению различных образовательных мероприятий направленных на получение знаний.

Активное развитие информационных технологий отразилось и на системе образования. Они стали неотъемлемой частью любого образовательного учреждения и стали незаменимым средством в организации и управления образовательным процессом.

Учебный процесс - целенаправленное взаимодействие преподавателя и учащихся, в ходе которого решаются задачи образования, развития и воспитания учащихся. Цель данного процесса в высшем учебном заведении – подготовка специалиста нужной квалификации, которая определяется как государственным образовательным стандартом, так и требованиями рынка. Для достижения поставленной цели необходимо, в первую очередь, изучить проблему управления учебным процессом и определить его специфику

Цель работы: разработать обучающую систему для дисциплины IT-направления.

Объект работы: учебный процесс подготовки IT-специалиста.

Предмет исследования: процесс обучения курсу «Разработка программных приложений».

Исходя из цели работы необходимо решить следующие задачи:

- исследовать существующие IT аналоги;
- проанализировать процесс обучения курсу «Разработка программных приложений» и выявить аспекты, которые нуждаются в модернизации;

- сформулировать требования к АИС;
- спроектировать базу данных;
- составить техническое задание.

В ходе курсовой работы должна быть спроектирована обучающая информационная система по дисциплине “Разработка программных приложений”.

Методы исследования:

- структурный подход к анализу и проектированию ИС;
- методология объектно-ориентированного анализа и проектирования ИС;
- методология онтологического подхода к моделированию ИС;
- методология объектно-структурного подхода к моделированию ИС;
- CASE-технологии структурного и объектно-ориентированного анализа и проектирования.

В первой главе происходит анализ предметной области, проводится сравнительный анализ современных методологий и технологий концептуального моделирования информационных систем. В соответствии с концепцией реинжиниринга на основе методологии IDEF0 строится модель процесса «КАК ЕСТЬ» (AS-IS) и проводится ее анализ и декомпозиция отдельных процессов.

Во второй главе описываются основные модели будущей системы. На основе методологии DFD строится развернутая модель учебного процесса «КАК ДОЛЖНО БЫТЬ» (TO-BE). Выполняется декомпозиция модернизированных процессов модели. Формулируются требования к новой АИС.

Для лучшего представления об объекте автоматизации проводится анализ известных ИТ-решений АИС на предмет соответствия сформулированным в разделе требованиям известных IT решений. В конце данной главы приводится обоснование решения о разработке новой АИС и логическое моделирование для уточнения основных выводов из ее концептуальной модели и постановки задачи на разработку специфического программного обеспечения.

В третьей главе описывается практическая реализация обучающей системы.

В заключении подведены итоги исследования, сформированы окончательные выводы по рассматриваемой теме.

Глава 1 Анализ предметной области автоматизации процесса обучения по курсу «Разработка программных приложений»

1.1 Выбор технологии анализа проектирования

Проектирование будущей информационной системы начинается с выбора методологии проектирования.

Рассмотрим основную информацию по методологии SADT.

Методология SADT представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели объекта какой-либо предметной области.

Правила SADT включают:

- графическое представление функции в виде блока;
- связи между функциями и внешней средой отображаются с помощью стрелок.

Моделирование системы с помощью IDEF0 дает наиболее полное представление о ее функционировании и движении потоков в рамках этой системы.

Модель DFD описывает систему с позиции обработки в ней информации.

На основе имеющихся данных построим таблицу 1.1 для сравнительного анализа методологий и выбора наиболее подходящей.

Таблица 1.1 - Сравнительный анализ методов проектирования

Критерий сравнения	IDEF0	DFD
1. Представление технологических процессов	+	-
2. Представление информационных процессов	+	+
3. Декомпозиция процессов	+	+
Оценка	3	2

Таким образом, наиболее подходящим вариантом является проектирование системы с использованием методологии IDEF0.

1.2 Анализ существующих технологий обучения

Для обоснования необходимости использования обучающей системы необходимо проанализировать существующие подходы к системе обучения и выявить процессы, нуждающиеся в автоматизации.

Вопросы:

Как осуществляется процесс обучения по курсу «Разработка программных приложений» ?

Что у обучаемого вызывает особые затруднения при прохождении данного курса?

На рисунке 1.1 представлена контекстная диаграмма процесса обучения по курсу “Разработка программных приложений”.

В качестве механизма используется преподаватель.

Входными данными здесь являются цели и задачи курса, а обучаемый является участником процесс обучения.

В управлении задействованы:

- рабочая программа курса;
- критерии оценивания знаний обучаемого.

Рабочая программа включает в себя:

- введение в курс (описание предмета, актуальности, цели и задач изучения курса);
- календарно-тематический план (план изучения лекций, выполнения практических работ, график сдачи тестов и итогового контроля);
- методические рекомендации по самостоятельному изучению курса;
- описание системы контроля успеваемости;
- сведения о преподавателе-авторе курса.

Потоками механизмов системы являются:

- преподаватель;

- обучаемый

Выходными данными служит оценивание знаний обучаемого.

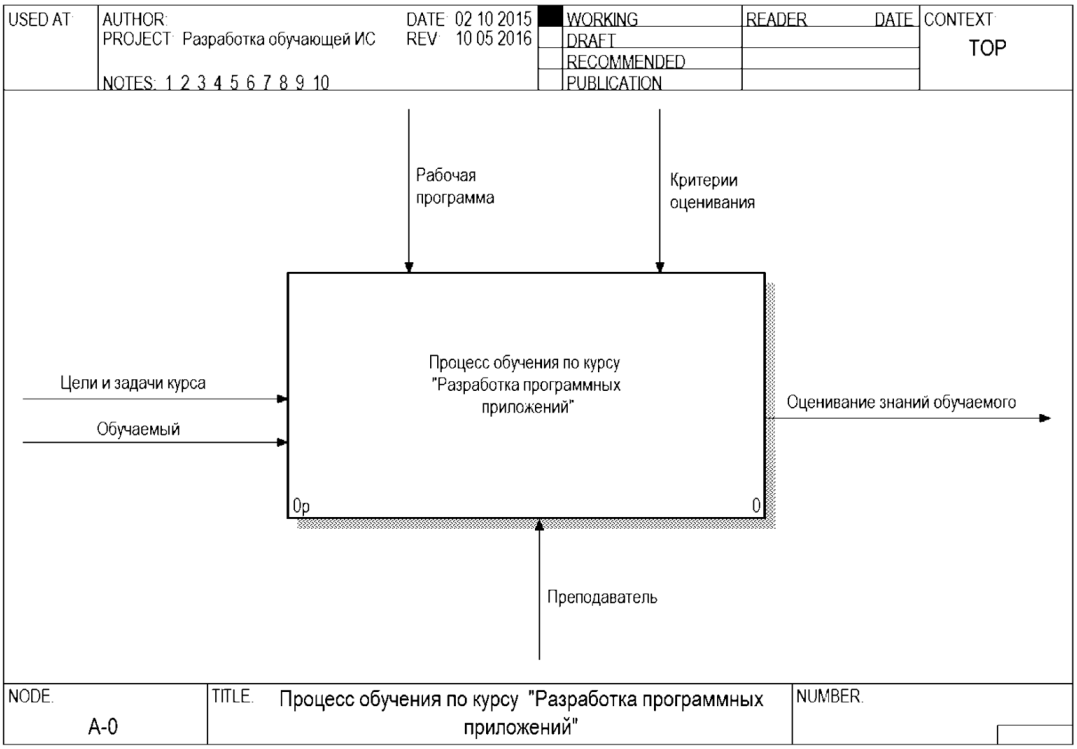

Рисунок - 1.1 - Функционально-ориентированная модель деятельности «Процесс обучения по курсу «Разработка программных приложений»» (IDEF0)

В декомпозиции контекстной диаграммы (Рис. 1.2) процессы разбиваются на следующие подпроцессы:

- введение в технологию разработки прикладных программ;
- разработка Windows-приложения в среде Delphi и основные приемы визуального программирования;
- разработка расчетных программ;
- введение в технологию проектирования;
- тестирование программного обеспечения;
- проверка знаний обучаемого.

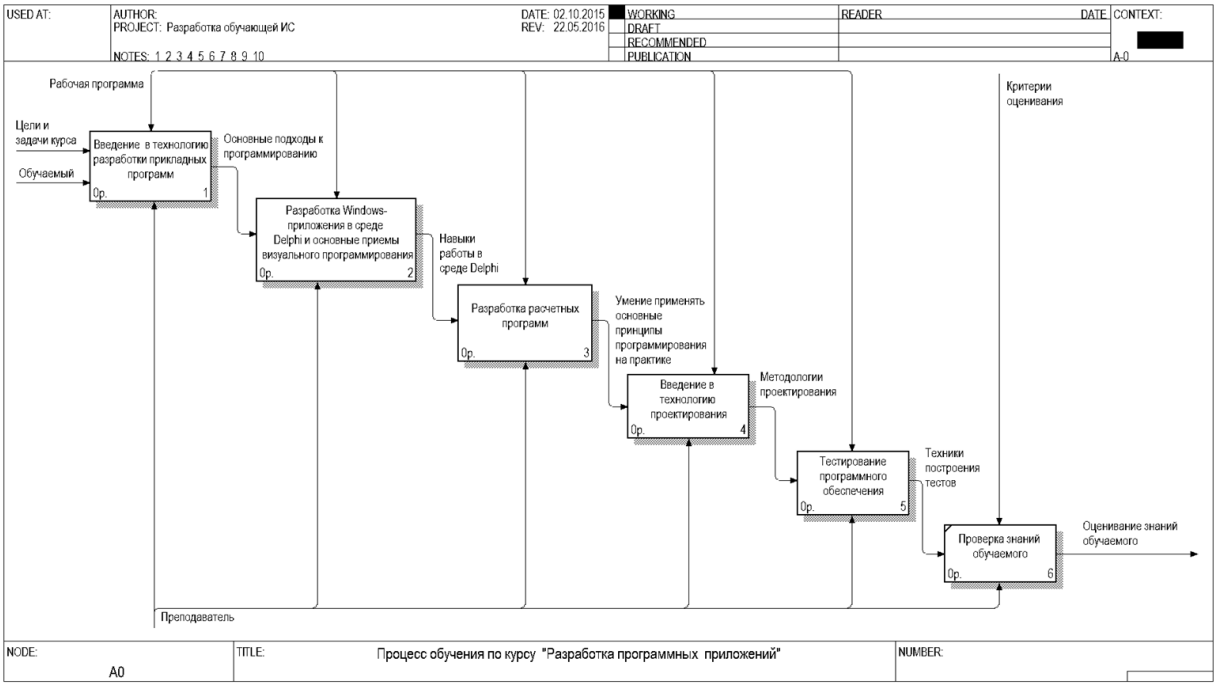

Рисунок 1.2 - Функционально-ориентированная модель декомпозиции контекстной диаграммы «Процесс обучения по курсу «Разработка программных приложений»» (IDEF0)

В процессе «Разработка расчетных программ» (Рис. 1.3) показана разработка простых программ на вход, которым подаются какие-то исходные данные, а на выходе выводятся результаты расчетов.

Весь представленный перечень задач связан с решением типичных задач по программированию и представляет собой вводную практическую часть.

Рассмотрим более процессы, представленные на рисунке 1.3:

1. «Введение в технологию программирования» - знакомство с основными подходами к программированию.
2. «Разработка Windows приложения в среде Delphi и основные приемы визуального программирования» - разработка программного приложения в среде Delphi и знакомство с технологией визуального программирования (технологией программирования с использованием готовых графических объектов).
3. «Разработка расчетных программ» - разработка программ для выполнения математических расчетов.

4. «Введение в технологию проектирования» подразумевает ознакомление с существующими технологиями проектирования и непосредственно технологический процесс проектирования с использованием определенной методологии проектирования.
5. «Проверка знаний обучаемого» - проведение контрольных работ для проверки знаний обучаемого по изученному материалу.

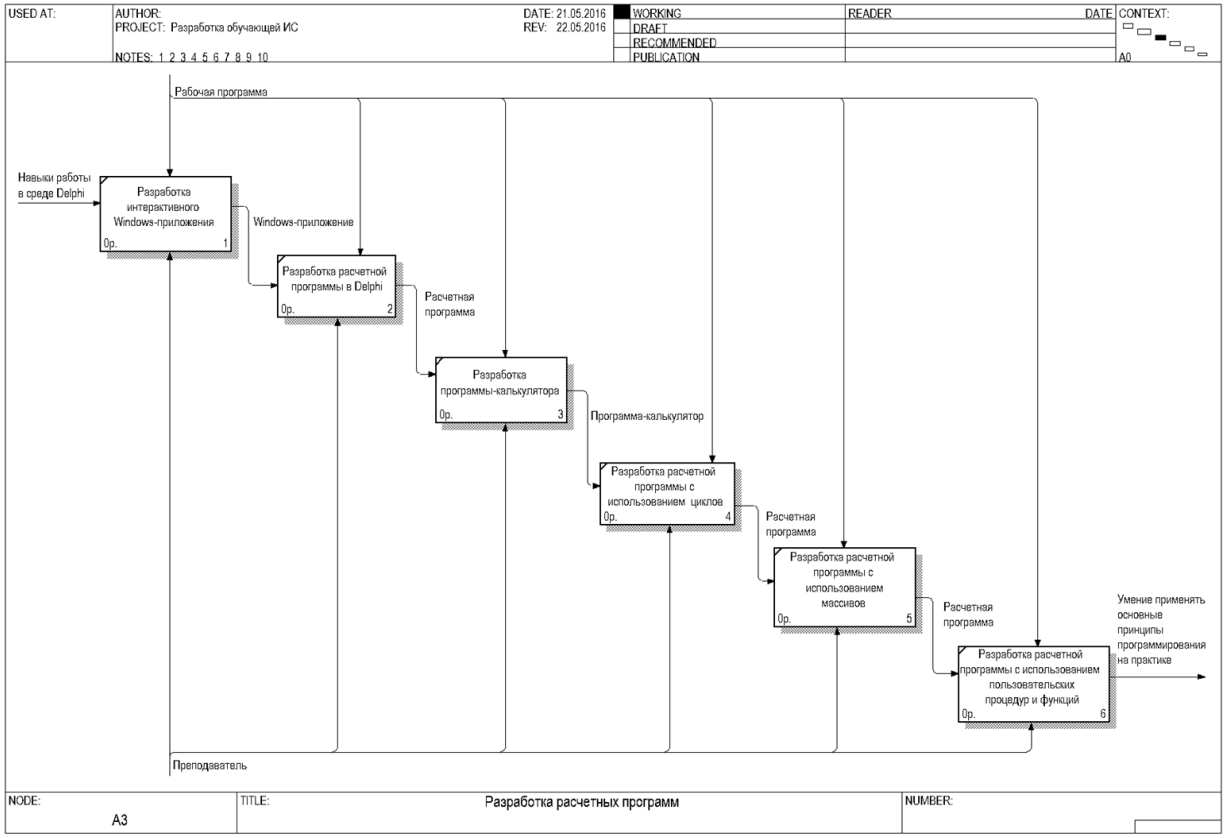

Рисунок 1.3 - Функционально-ориентированная модель декомпозиции процесса «Разработка расчетных программ» (IDEF0)

В декомпозиции процесса «Тестирование программного обеспечения» (Рис. 1.4) показана теоретическая и практическая часть по тестированию программного обеспечения.

Преподаватель предоставляет общие сведения о технологии тестирования обучаемым.

Обучаемый на основе этой информации выполняет практическое задание, связанное с поиском ошибок в программе “ListBoxer”.

Следующим этапом является составление обучаемым тестовой документации.

Составление тестовой документации происходит на основе тестовых требований, которые составляет обучаемый.

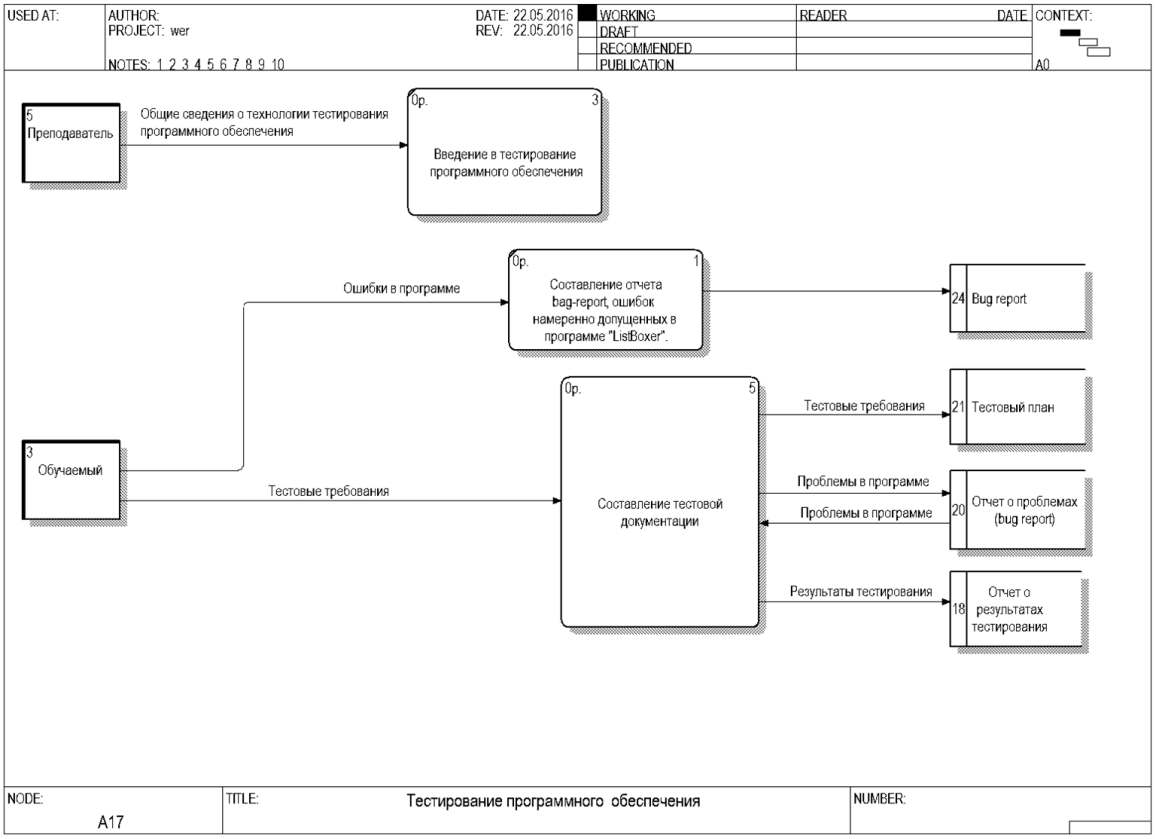

Рисунок 1.4 - Функционально-ориентированная модель декомпозиции процесса «Тестирование программного обеспечения» (DFD)

В декомпозиции процесса «Составление тестовой документации» (Рис. 1.5) показаны основные этапы составления тестовой документации:

- составление тестового плана;
- тестирование программы;
- подготовка наборов тестовых данных;
- составление отчета о результатах тестирования.

Составление тестового плана производится на основе тестовых требований.

В процессе тестирования программы происходит поиск дефектов в программе и на его основе формируется отчет о проблемах (bug report).

На основе отчета о проблемах производится подготовка тестовых данных и составление отчета о результатах тестирования.

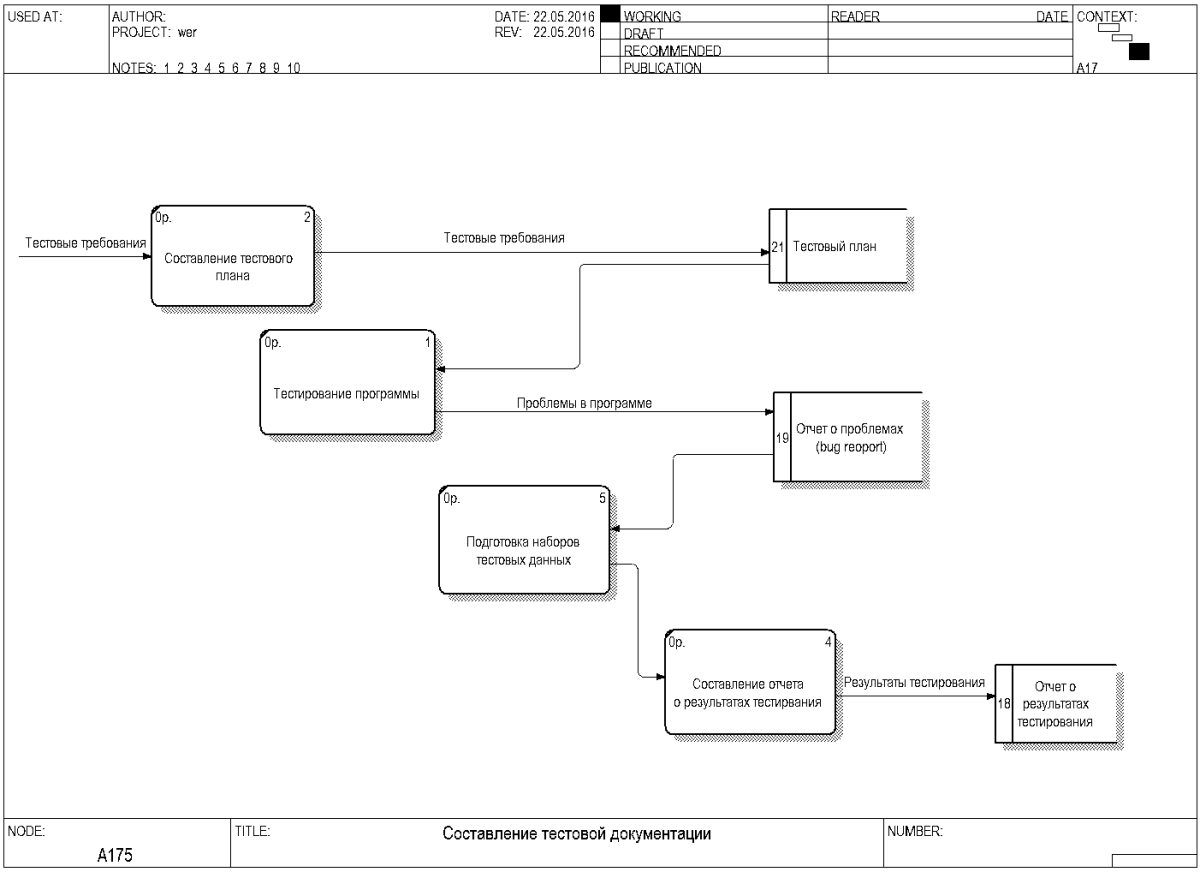

Рисунок 1.5 - Функционально-ориентированная модель декомпозиции процесса «Составление тестовой документации» (DFD)

Таким образом, была отражен существующий процесс обучения разработке программных приложений.

1.3 Выявление недостатков существующего процесса обучения и рекомендации по его усовершенствованию с помощью ИТ.

Исходя из анализа существующих технологий, были выделены следующие недостатки существующего процесса обучения:

- недостаточная интерактивность процесса обучения;
- отсутствие единого централизованного источника информации;

- недостаточная наглядность процесса разработки программ, что повышает трудоемкость при выполнении обучаемым практических заданий.

Была построена контекстная диаграмма «КАК ДОЛЖНО БЫТЬ» (TO BE) представленная на рисунке 1.6.

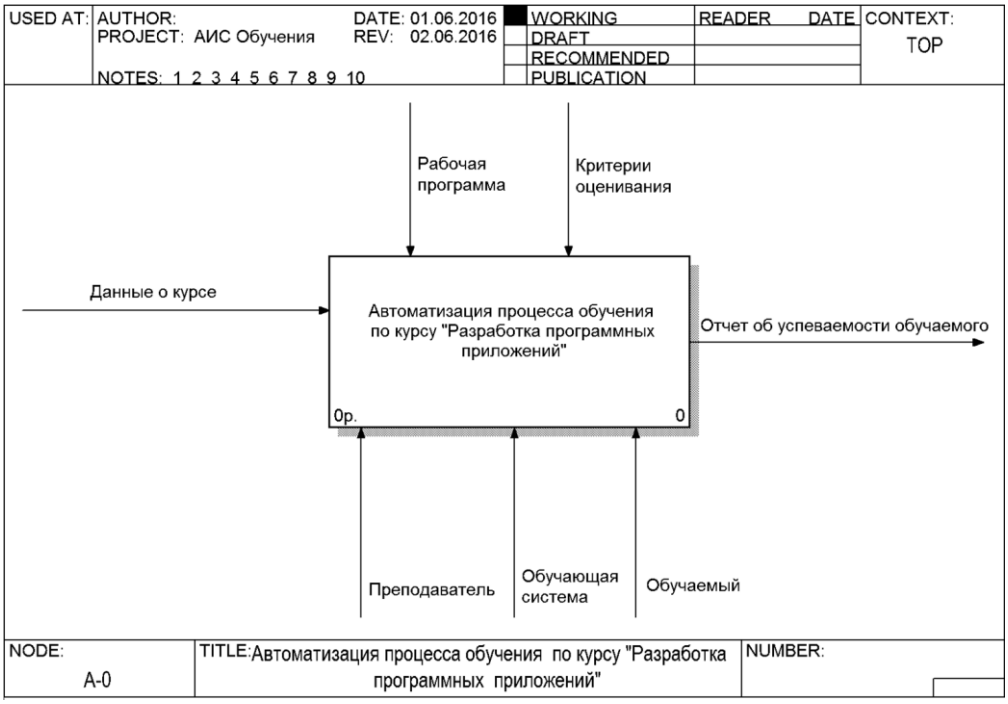

Рисунок 1.6 - Процесс «Автоматизация процесса обучения по курсу «Разработка программных приложений»»

Если сравнивать контекстную диаграмму «КАК ДОЛЖНО БЫТЬ», представленную на рисунке 1.6 с диаграммой «КАК ЕСТЬ», изображенной на рисунке 1.1, можно заметить, что механизмами являются преподаватель и обучающая система.

На диаграмме, представленной, на рисунке 1.7 показана декомпозиция процесса «Автоматизация процесса обучения по курсу «Разработка программных приложений»» (Рис. 1.7).

Описание целей и задач описывается на основании рабочей программы.

В результате прохождения курса подсчитываются результаты прохождения курса и выводятся результаты.

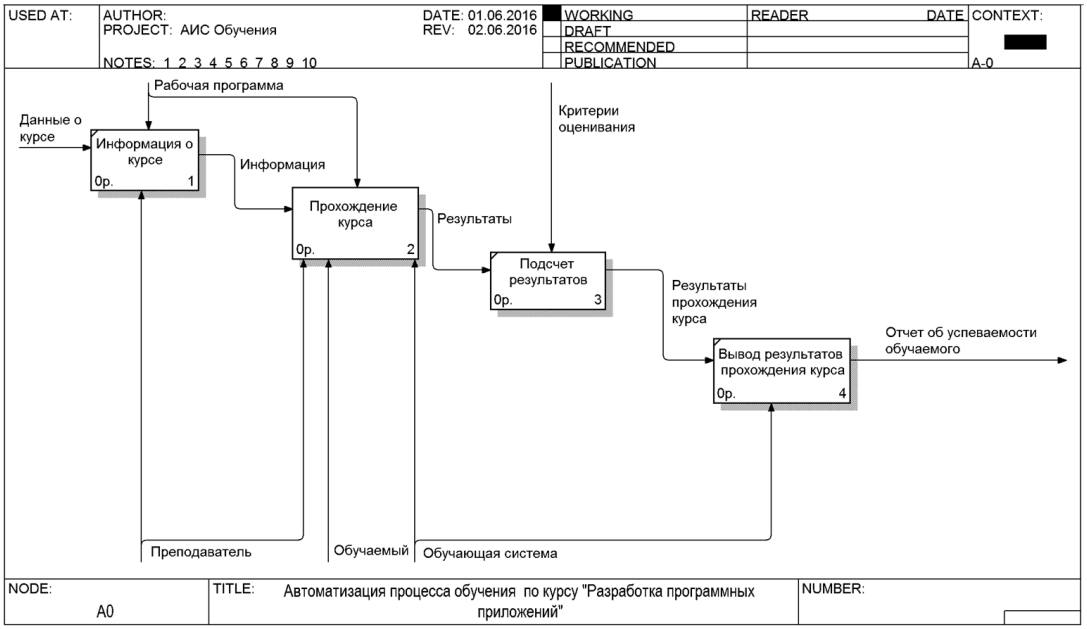

Рисунок 1.7 - Функционально-ориентированная модель декомпозиции процесса «Автоматизация процесса обучения по курсу «Разработка программных приложений»»

В процессе «Прохождение курса» (Рис. 1.8) отображена последовательность прохождения данного курса.

Опишем более детально процессы, изображенные на рисунке 1.8:

1. «Формирование теоретического материала» - на данном этапе преподаватель производит ввод теоретического материала по курсу в обучающей системе для определенного курса.
2. «Предоставление теоретического материала в виде презентации» - на данном этапе обучаемый изучает теоретический материал в виде презентации в обучающей системе с возможностью перехода от одного фрагмента к другому с возможным наличием интерактивных элементов для повышения наглядности процесса усвоения учебного материала.
3. «Формирование заданий» - на данном этапе преподаватель занимается планированием практической части курса, добавляя для созданного курса тесты, задания во встроенном в обучающую систему редакторе кода.
4. «Выполнение заданий во встроенном редакторе кода» - на данном этапе обучаемый выполняет созданные преподавателем задания в редакторе

кода, в котором реализован механизм всплывающих подсказок в случае получения неверного результата после выполнения кода.

5. «Прохождение тестирования по изученному материалу» - процесс проведения тестирования с целью проверки знаний обучаемого.

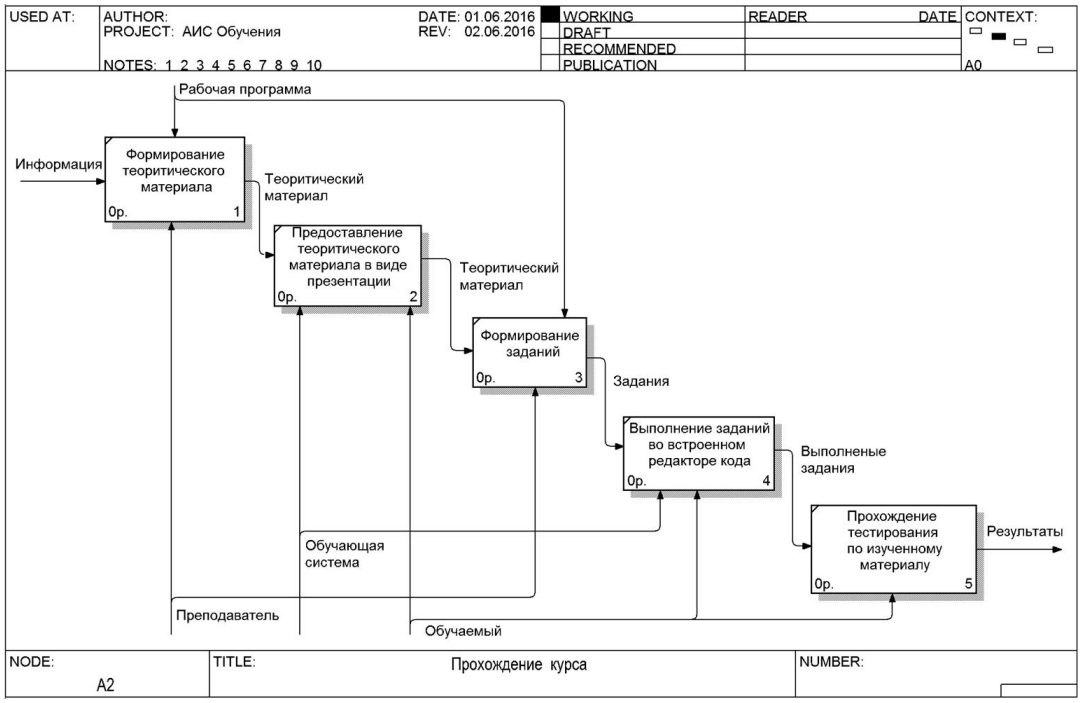

Рисунок 1.8 - Процесс «Прохождение курса»

Преподаватель на основе рабочей программы курса занимается формированием теоретического материала и его предоставлением в виде презентации, а также формированием практических заданий.

Обучаемый занимается выполнением заданий составленных преподавателем во встроенным в обучающую систему редакторе кода и проходит тестирование по изученному материалу

Таким образом, применение средств электронного обучения необходимо для повышения качества и эффективности существующих методов преподавания основам разработки прикладных программ.

Вывод по первой главе

В результате проведения анализа предметной области было произведено сравнение методологий и технологии концептуального моделирования АИС.

Глава 2 Разработка новой технологии обучения

2.1 Разработка требований

Определим основные требования, предъявляемые к будущей системе. Требования к системе описаны по методологии FURPS+(таблица 2.1).

Классификация требований к системе FURPS+ была разработана Робертом Грэйди (Robert Grady) из Hewlett-Packard. Сокращение FURPS расшифровывается так:

- functionality, функциональность;
- usability, удобство использования;
- reliability, надежность;
- performance, производительность;
- supportability, поддерживаемость;
- + необходимо помнить о возможных ограничениях;

Таблица 2.1 – Требования к системе

ID требования	Требование	Статус	Полезность	Риск	Стабильность	Целевая версия
Functionality — Функциональные требования						
1.	Обучение фундаментальным принципам программирования в форме презентации	Одобренные	Критичное	Средний	Низкая	1.0.0.0
2.	Реализация интерактивного обучения с использованием встроенного в систему редактора кода с выводом наводящих подсказок при запуске компиляции кода пользователем	Одобренные	Критичное	Средний	Низкая	1.0.0.0

Продолжение таблицы 2.1

3.	Реализация тестирования по пройденному материалу	Одобренные	Критичное	Средний	Низкая	1.0.0.0
Usability — Требования к удобству использования						
4.	Разрешение экрана 1024x1200	Одобренные	Критичное	Низкий	Низкая	1.0.0.0
Reliability — Требования к надежности						
5.	Доступ посетителю 24 часа в сутки	Одобренные	Критичное	Низкий	Средний	1.0.0.0
6.	Матричное разграничение доступа	Одобренные	Критичное	Низкий	Средний	1.0.0.0
7.	Защита от несанкционированного доступа	Предложенные	Критичное	Средний	Средняя	1.0.0.0
Performance — Требования к производительности						
8.	Время реакции системы на события должно быть не более двух секунд	Предложенные	Критичное	Средний	Средняя	1.0.0.0
Supportability — Требования к поддержке						
9.	Время устранения возникших проблем 10 мин	Предложенные	Критичное	Средний	Средняя	1.0.0.0
10.	Обновление системы бесплатно и происходит автоматически	Предложенные	Критичное	Средний	Средняя	1.0.0.0
Ограничения реализации, разработки, построения, написания программного кода						
11.	Управление БД осуществляется через СУБД MySQL	Предложенные	Критичное	Средний	Средняя	1.0.0.0
12.	Данные хранятся в виде реляционных таблиц базы данных сайта	Одобренные	Критичное	Низкий	Низкая	1.0.0.0

Продолжение таблицы 2.1

Требования к интерфейсам						
13.	Интерактивные элементы обучения	Предложенные	Критичное	Средний	Средний	1.0.0.0
14.	Для реализации анимации должен быть использован язык HTML 5	Предложенные	Критичное	Средний	Средний	1.0.0.0

Таким образом, на этапе анализа и выработки требований к новой системе было получено:

- система должна реализовывать 3 функциональных требований
- система должна реализовывать 11 нефункциональных требований

Все вышеперечисленные требования учитывались при разработке системы

2.2 Логическое моделирование

Для определения цели модели, необходимо составить список вопросов, на которые должны отвечать модель. Для этого нужно сформулировать множество вопросов и выбрать основные:

1. Кто отвечает за формирование учебного процесса?
2. Кто является участником учебного процесса?

В качестве точки зрения следует выбрать точку зрения преподавателя, так как с этой точки зрения проще всего ответить на поставленные вопросы.

Исходя из вышеперечисленного, можно сформулировать следующую цель моделирования: определить, каким образом будет организован учебный процесс.

Для данной предметной области выделим следующих актеров:

- преподаватель;
- обучаемый;
- администратор системы.

На основании вышеперечисленного можно выделить следующие прецеденты, представленные в таблице 2.2, которые должны быть реализованы в обучающей системе.

Таблица 2.2 – Краткое описание прецедентов

Прецеденты	Краткое описание
1	2
Ввод теоретического материала	Ввод теоретического материала курса
Добавление курса	Добавление курса преподавателем
Формирование заданий	Создание заданий различных типов исходя из целей и задач курса.
Изучение теоретического материала	Изучение теоретического материала
Изучение теоретического материала	Изучение теоретического материала по определенному курсу
Выполнение заданий во встроенном редакторе кода	Выполнение заданий во встроенном в систему редакторе кода со всплывающими подсказками в случае неверных действий обучаемого.
Прохождение тестирования	Прохождение тестирования для проверки знаний обучаемого
Добавление пользователя в систему	Добавление новых пользователей в системы с целью дальнейшего разграничения прав доступа к данным
Проверка прав пользователя	Проверка разграничения прав доступа пользователя к данным
Настройка прав доступа к системе	Настройка прав доступа пользователей к данным в соответствии с заданной политикой информационной безопасности

Разработанная диаграмма вариантов использования для основных прецедентов проектируемой системы представлена на рисунке 2.1.

На диаграмме вариантов использования представлен функциональный аспект разрабатываемой системы.

Преподаватель занимается вводом теоретического материала по курсу, формированием заданий и созданием курса

Обучаемый изучает теоритический материал после чего выполняет задания в редакторе кода и проходит тестирование по изученному материалу.

Рисунок 2.1 - Диаграмма вариантов использования

Далее в таблицах 2.3 – 2.7 представлены спецификации основных прецедентов.

Таблица 2.3 - Описание прецедента

Прецедент: Добавление курса
ID: 1
Краткое описание: Добавление курса преподавателем
Главные актеры: Преподаватель
Второстепенные актеры:

Продолжение таблицы 2.3

Предусловие: Прецедент начинается по инициативе преподавателя
Основной поток: 1. Добавление обучающего курса
Постусловие: Созданный курс
Альтернативные потоки: Нет

Таблица 2.4 - Описание прецедента

Прецедент: Ввод теоретического материала курса
ID: 2
Краткое описание: Ввод теоретического материала по курсу
Главные актеры: Обучаемый
Второстепенные актеры:
Предусловие: Прецедент начинается по инициативе преподавателя
Основной поток: 1. Ввод теоретического материала по курсу
Постусловие: Вывод отчета о результатах
Альтернативные потоки: Нет

Таблица 2.5 - Описание прецедента

Прецедент: Формирование заданий
ID: 3

Продолжение таблицы 2.5

Краткое описание: Изучение теоретического материала, выполнение контрольных заданий
Главные актеры: Обучаемый
Второстепенные актеры:
Предусловие: Прецедент начинается по инициативе пользователя
Основной поток: 1. Добавление заданий в редакторе кода 2. Добавление теста
Постусловие: Вывод отчета о результатах
Альтернативные потоки: Нет

Таблица 2.6 - Описание прецедента

Прецедент: Изучение теоретического материала
ID: 4
Краткое описание: Изучение теоретического материала, выполнение контрольных заданий
Главные актеры: Обучаемый
Второстепенные актеры:
Предусловие: Прецедент начинается по инициативе пользователя
Основной поток: 1. Изучение теоретического материала предоставленного обучаемого в виде презентации

Продолжение таблицы 2.6

Постусловие: Вывод отчета о результатах
Альтернативные потоки: Нет

Таблица 2.7 - Описание прецедента

Прецедент: Выполнение заданий во встроенном редакторе кода
ID: 5
Краткое описание: Выполнение последовательности заданий относящихся к конкретному курсу
Главные актеры: Обучаемый
Второстепенные актеры:
Предусловие: Прецедент начинается по инициативе обучаемого
Основной поток: 1. Выполнение заданий в редакторе кода 2. Прохождение теста
Постусловие: Вывод отчета о результатах
Альтернативные потоки: Нет

Таким образом, были разработаны основные технологические этапы, которые необходимо реализовать для автоматизации существующей технологии учебного процесса разработке программных приложений.

На основе диаграммы вариантов использования была построена диаграмма классов (Рис. 2.2) отображающая классы, их атрибуты и операции, а также связи между классами с указанием кратности.

Классами здесь выступают *Teacher*, *Course*, *Task*, *Learner*, *Progress и* дочерние от класса *Task* классы *Lab*, *Quiz* и *CodeEditor*.

Все классы содержат функции установки и получения значений (set и get) в зависимости от функционального назначения данного класса.

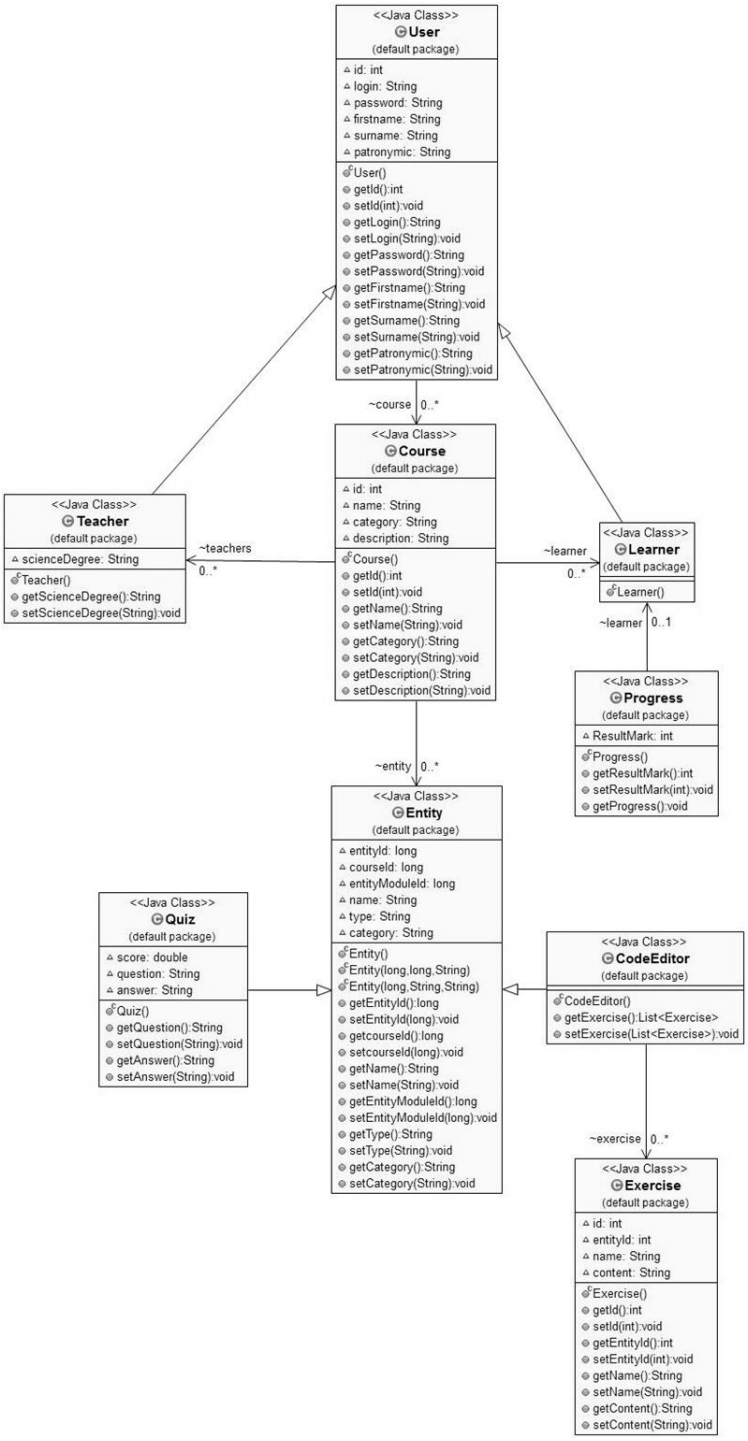

Рисунок 2.2 - Диаграмма классов

Участниками процесса обучения в электронной среде являются: User – родительский класс для определения абстрактного пользователя, который, в

свою очередь, делится на Learner – обучаемый, и Teacher – преподаватель.

Класс *Entity* содержит название курса, категорию и описание курса,

Класс *Quiz* (*Тест*) дополнительно включает в себя переменные вопрос тестирования и ответ на вопрос тестирования.

Класс *CodeEditor* содержит список заданий.

Класс *Progress (Успеваемость)* включает общую оценку за курс и операцию getProgress() выводящая успеваемость определенного обучаемого.

Логическая модель данных, изображенная на рисунке 2.3, была построена на основе диаграммы классов.

Она состоит из пяти сущностей: *Teacher*, *Course*, *Learner*, *Entity*, *Quiz*, *CodeEditor*, *Exercise* и *Progress*.

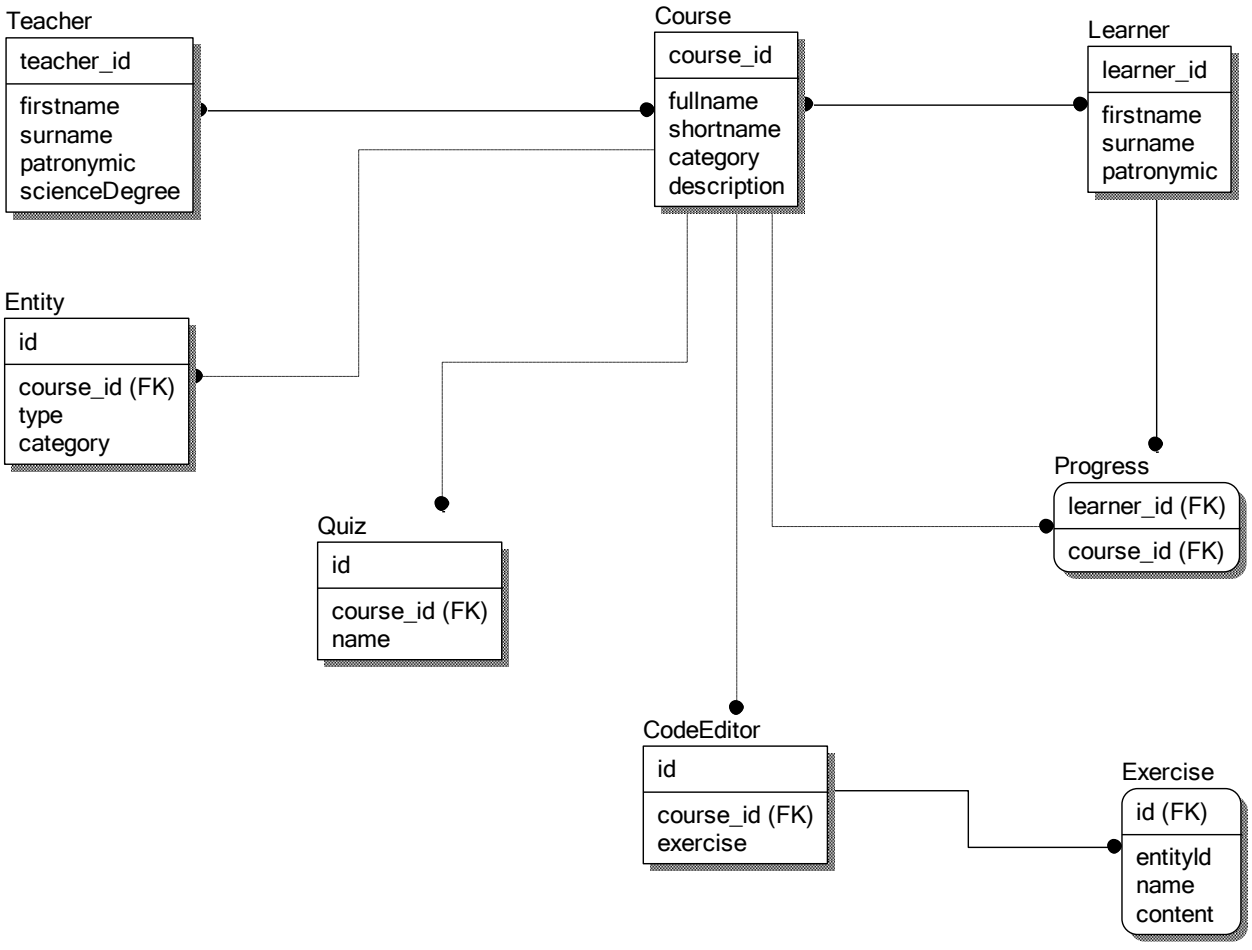

Рисунок 2.3 – Логическая модель данных

Каждая сущность имеет свои атрибуты и методы. Сущности могут быть связаны между собой связями различной кратности [11, 154].

Так связь “много-ко-многим” между сущностями *Course* и *Teacher* указывает на то, что много преподавателей могут вести несколько курсов.

Курс может состоять из нескольких заданий разного типа, о чем говорит неидентифицирующая связь “один-ко-многим” между сущностями *Course* и *Task*.

Связь “много-ко-многим” между сущностями *Learner* и *Course* предусматривает, что много студентов могут записаться на прохождение сразу нескольких курсов.

В соответствии с вышеупомянутым между сущностями *Course* и *Progress* связь имеет кратность “один-ко-многим”, потому что у каждого ученика есть успеваемость по каждому курсу своя.

Поскольку каждый отдельно взятый ученик может просматривать только свою успеваемость, то связь между сущностями *Learner* и *Progress* имеет кратность “один-к-одному”, но без ученика успеваемость существовать не может, поэтому связь идентифицирующая.

Курс может содержать несколько заданий различного типа.

Практическое задание может состоять из нескольких вопросов, но в свою очередь вариантов ответов на один вопрос может быть несколько как, например, при множественном выборе в вопросах тестирования.

2.3 Физическое моделирование.

Физическая модель данных определяет способ фактического хранения данных в запоминающих устройствах памяти. В ней рассматриваются отношения, индексы и кластеризация данных для повышения эффективности обработки данных.

На рис. 2.4 представлена физическая модель данных отражающая способы размещения данных в среде хранения и способы доступа данным.

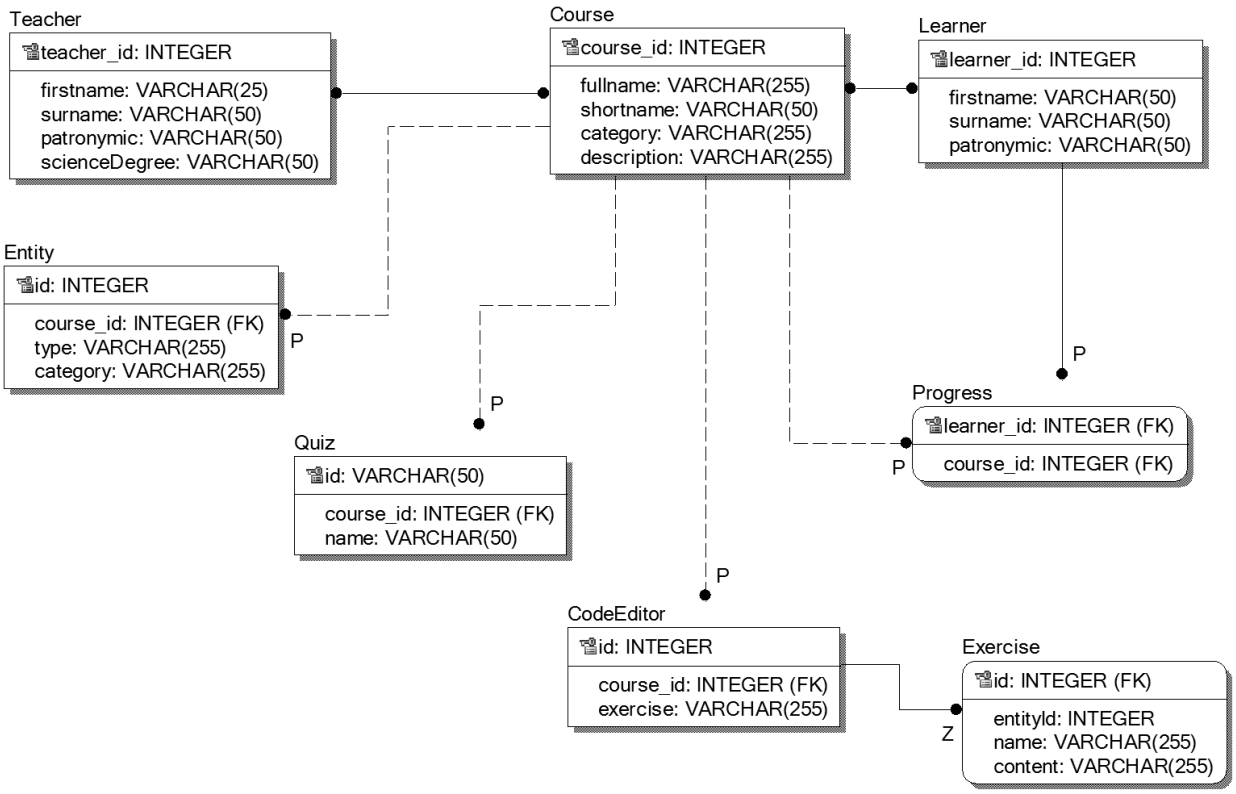

Рисунок 2.4 – Физическая модель данных

Таким образом, была построена физическая модель данных отражающая способы размещения данных в среде хранения и способы доступа данным

2.4 Разработка архитектуры системы.

Архитектура приложений для платформы Java EE разделена на уровни: уровень клиента, промежуточный уровень (состоящий из веб-слоя и бизнес-слоя) и уровень корпоративной информационной системы (Enterprise Information System, EIS) [10, 41].

Уровень клиента представляет собой браузер, подключающийся к серверу Java EE посредством протокола передачи гипертекста HTTP или любое приложение на любой клиентской машине [10, 42].

На уровне доступа к данным или промежуточном уровне размещается сервер Java EE [10, 43].

Веб-слой служит для управления обменом данными между уровнем клиента и бизнес слоем [10, 43].

Бизнес-слой выполняет бизнес-логику, решая задачи или удовлетворяя конкретные потребности в коммерческих проектах с использованием данных из базы данных, размещенной на уровне EIS [10, 43].

Уровень EIS или уровень данных состоит из блоков хранения данных. На нем размещена система управления базами данных для хранения данных о пользователях, их успеваемости, курсах и. т. д [10, 45].

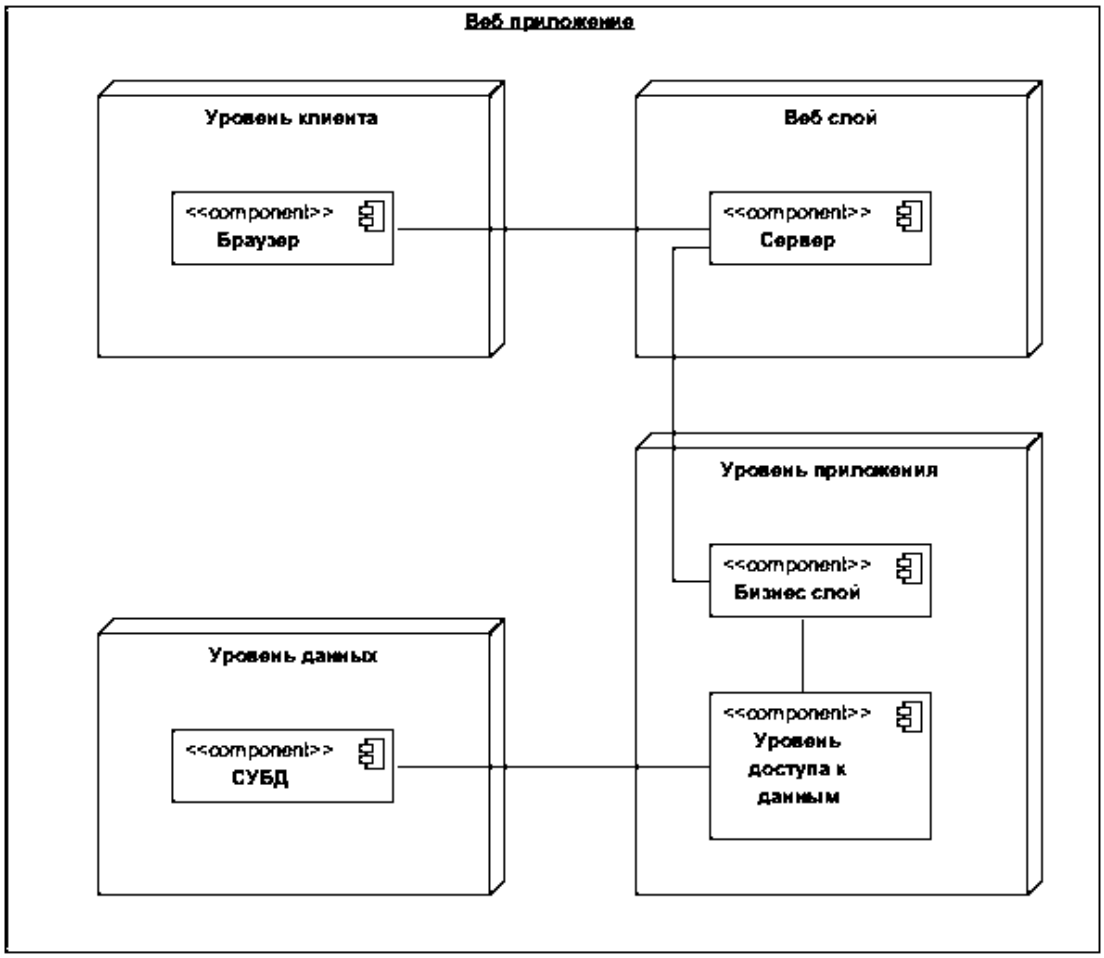

Рисунок 2.5 – Диаграмма развертывания

Таким образом, была построена диаграмма развертывания для отображения архитектуры создаваемой системы.

2.5 Структурная схема взаимосвязи между логическими и физическими модулями.

На диаграмме компонентов (Рис. 2.6) показаны взаимосвязи между логическими и физическими модулями, из которых состоит моделируемая система [11, 44].

Для абстрагирования и инкапсулирования доступа к источнику данных был использован DAO (Data Access Object) – паттерн для доступа к данным [10, 172].

На уровне данных представлена реализация интерфейса CRUD для соответствующих классов в виде интерфейсов UserDAO, CourseDAO и EntityDAO при помощи паттерна Factory [10, 172].

бизнес-логику, заботятся о транзакциях и безопасности [1 ,282].

Уровень представления отвечает за отображение данных.

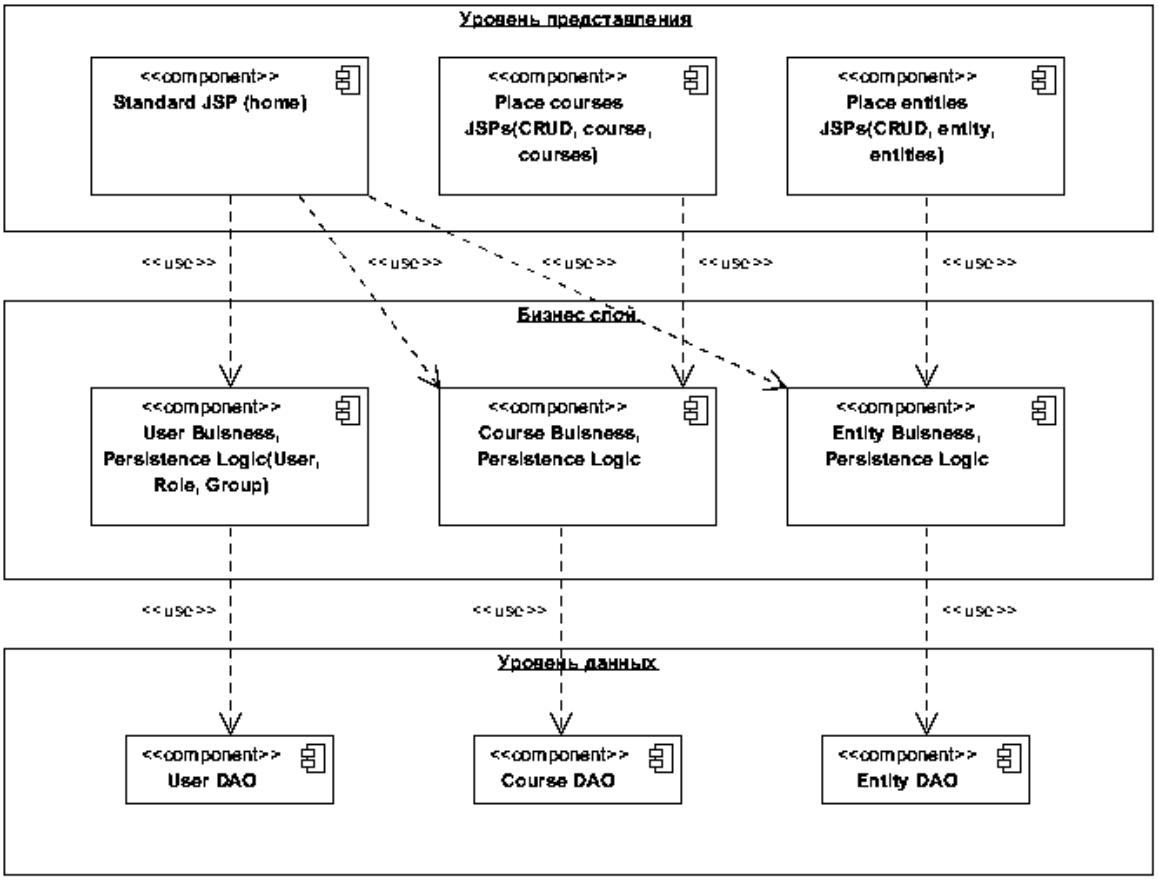

Рисунок 2.6 – Диаграмма компонентов

Бизнес слой содержит классы User, Course и Entity – три EJB (Enterprise JavaBeans) компонента для реализации бизнес логики приложения [1 ,283].

Вывод по второй главе

Построена концептуальная модель потоков данных учебного процесса и на ее основе был сделан вывод о том, какие функциональные требования предъявляются к обучающей АИС.

Глава 3 Практическая реализация обучающей системы по курсу «Разработка программных приложений»

3.1 Обоснование использования программных средств разработки

В процессе разработки программного обеспечения используются интегрированные среды разработки (IDE) с целью повышения удобства написания программного кода.

Существует множество инструментов программирования со своими достоинствами и недостатками.

Современные IDE должны обеспечивать:

- быстрое интеллектуальное редактирование кода;
- легкое и эффективное управление проектами;
- быструю разработку пользовательского интерфейса;
- создание кода без ошибок;
- поддержка нескольких языков программирования;
- поддержка нескольких платформ [20].

Для разработки web приложений выбрана технология Java Enterprise Edition (Java EE).

Технология Java EE является расширением языковой платформы Java, которое позволяет создавать масштабируемые, мощные и переносимые корпоративные приложения [20].

Для выбора среды разработки необходимо провести сравнительный анализ существующих сред разработки.

Рассмотрим основную информацию по нескольким наиболее популярным IDE для разработки программ на языке Java:

- NetBeans IDE;
- Eclipse IDE;
- IntelliJ IDEA;

IntelliJ IDEA в отличие от NetBeans и Eclipse производит автосохранение, что значительно упрощает процесс написания программы из-за отсутствия

необходимости в постоянном сохранении внесенных в проект изменений и автокомпиляцию кода, а гибкое управление локальной историей автоматически сохраненных изменений позволяет просматривать программный код до и после редактирования. IntelliJ IDEA доступна в двух версиях: полностью свободной версией Community Edition и платной Ultimate Edition, в которой реализована поддержка Java EE, UML-диаграмм.

NetBeans и Eclipse IDE являются полностью свободно распространяемым программным обеспечением с поддержкой всех новейших технологий Java.

Eclipse IDE производит автокомпиляцию только после сохранения внесенных изменений произведенного вручную.

В NetBeans IDE редактор форм для визуального программирования, в то время как в Eclipse IDE он подключается в виде дополнительно подключаемого модуля, который необходимо загрузить. Так же в NetBeans IDE встроен сервер Glassfish для быстрой сборки и развертывания веб-приложения на данный сервер [20].

Каждая из рассмотренных сред разработки обеспечивает удобный и быстрый способ написания кода для программирования и выбор.

На основе имеющихся данных приведена таблица 3.1 для сравнительного анализа сред разработки.

Таблица 3.1 - Сравнительный анализ сред разработки для программирования на языке Java

Критерии сравнения	NetBeans	Eclipse	IntelliJ IDEA
Открытое программное обеспечение	+	+	-

Продолжение таблицы 3.1

Технология визуального программирования	+	+	+
Автосохранение	-	-	+
Встроенный сервер приложений	+	-	+
Всего	3	2	3

Таким образом, в качестве наиболее подходящего инструмента разработки была выбрана NetBeans IDE, потому что она является бесплатной и обладает большей централизацией компонентов по сравнению с Eclipse.

3.2 Реализация функциональных требований

В процессе разработки обучающей системы были выполнены следующие задачи:

1. Реализация механизмов защиты компьютерной системы:

- *идентификация* (именование и опознавание), *аутентификация* (подтверждение подлинности) пользователей системы;
- *разграничение доступа* пользователей к ресурсам системы и авторизация (присвоение полномочий) пользователям [7, 29].

На рисунке 3.1 представлена реализация JavaBean класса SecurityBean, в котором производится инициализация интерфейсов используемых в системе, среди которых есть LocalUserDAO отвечающий за операции добавления, редактирования и удаления пользователей.

Компоненты JavaBeans – это многократно используемые классы Java, позволяющие разработчикам существенно ускорять процесс разработки WEB-приложений путем их сборки из программных компонентов. JavaBeans и другие компонентные технологии привели к появлению нового типа программирования – сборки приложений из компонентов, при котором разработчик должен

знать только сервисы компонентов; детали реализации компонентов не играют никакой роли.

```
public abstract class SecurityBean implements Serializable {
   /**
    * Session instance
    */
   @Inject
   protected HttpSession session;

   /**
    * Request instance
    */
   @Inject
   protected HttpServletRequest request;

   protected UserDAO userDAO;
   protected GroupDAO groupDAO;
   protected RightDAO rightDAO;
   protected CourseDAO courseDAO;
   {
      userDAO = new LocalUserDAO();
      groupDAO = new LocalGroupDAO();
      rightDAO = new LocalRightDAO();
      courseDAO = new LocalCourseDAO();
   }
   /**
    * Gets current user
    * @return current user
    */
   public User authorization()
   {
      return (User) session.getAttribute(USER_INSTANCE);
   }
}
```

Рисунок 3.1 – Реализация класса SecurityBean

На рисунке 3.2 представлена реализация класса AutentificationBean, который отвечает за аутентификацию.

```
public class AuthentificationBean extends SecurityBean {
    /**
     * Sets default user setting if user does not exist
     * @throws SecurityException
     */
    public void cheackAuthentification() throws SecurityException
    {
        if(authorization() == null) //User is Guest if not authentificated
        {
            logout();
        }
    }

    /**
     * Sets new user setting
     * @throws SecurityException
     */
    public void login() throws SecurityException
    {
        Map<String, String[]> parameters = request.getParameterMap();
        if(parameters.containsKey(USER_EMAIL_PARAM) && parameters.containsKey(USER_PASSWORD_PARAM))
        {
            String email = parameters.get(USER_EMAIL_PARAM)[0];
            email = email.trim();
```

```
        if(StringUtils.isEmpty(email))
        {
            throw new SecurityException("Enter email");
        }

        String password = parameters.get(USER_PASSWORD_PARAM)[0];
        if(StringUtils.isEmpty(password))
        {
            throw new SecurityException("Enter password");
        }
//          password = Encrypt.getHash(password);

        User user = userDAO.findByEmail(email);
        if(user.getEmail() != null)
        {
            if(user.getPassword().equals(password))
            {
                session.setAttribute(USER_INSTANCE, user);
            }
            else
            {
                throw new SecurityException("Invalid login or password");
            }
        }
        else
        {
            throw new SecurityException("Invalid login or password");
```

```
            }
        }
    }

    /**
     * Sets default user settings
     */
    public void logout()
    {
        session.setAttribute(USER_INSTANCE, LocalUserDAO.GUEST);
    }
}
```

Рисунок 3.2 – Реализация класса AuthentificationBean

Для удобства здесь были использованы строковые константы из класса SecurityConstants для проверки пользователя соответственно по введенному логину, паролю и электронной почте.

```
public final class SecurityConstants {
    /**
     * User instance in session
     */
    public final static String USER_INSTANCE = "user";

    /**
     * User email parameter name in authentification
     */
    public final static String USER_EMAIL_PARAM = "email";
```

```
    /**
     * User password parameter name in authentification
     */
    public final static String USER_PASSWORD_PARAM = "password";


}
```

Рисунок 3.3 – Реализация класса SecurityConstants

В случае прохождения успешной проверки пользователя производится авторизация пользователя вызовом функции “autorization” с использованием сессии.

Разграничение прав доступа происходит разделением пользователей на три группы:

1. Гость – не авторизовавшийся или не зарегистрированный пользователь (посетитель).
2. Пользователь – авторизовавшийся пользователь (преподаватель или обучаемый).
3. Администратор – специалист занимающийся обслуживанием сайта.

Оно обеспечивается посредством интерфейса LocalGroupDAO, где объявляются массивы класса Group с инициализацией переменных groupId, parentId, name.

Обучаемому доступны курсы для прохождения, а преподаватель может планировать формат, в котором будет проходить процесс обучения выстраивая подачу учебного материала по своему усмотрению.

Таким образом, обеспечивается разграничение прав пользователей.

Администратор системы реализует контроль функционирования системы, разграничивает права пользователей системы, выполняет резервное копирование базы данных системы.

Разделение групп пользователей показано на рисунке 3.4.

```
 * Group for anonymous users
 */
public static final Group GUESTS = new Group(0L, 0L, "Guests");

/**
 * Group for administrators
 */
public static final Group ADMINS = new Group(1L, 1L, "Admins");

/**
 * Group for anonymous users
*/
public static final Group USERS = new Group(2L, 2L, "Users");
```

Рисунок 3.4 – Разграничение прав доступа

Проверку на отношение к той или группе пользователи производится при помощи сервлета LoginServlet реализация которого представлена на рисунке 3.5.

```
public class LoginServlet extends HttpServlet {
CourseDAO courseDAO = DAOFacto-
ry.getDAOFactory(DAOFactory.DAOType.LOCAL).getCourseDAO();

   @EJB
   AuthentificationBean security;

   /**
    * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
    * methods.
    *
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    */
   protected void processRequest(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {

      try {
         security.login();
      } catch (SecurityException ex) {
         request.setAttribute("loginStatusText", ex.getMessage());
      }
```

```
        request.setAttribute("currentGroup", security.authorization().getGroupId());

        List<Course> courses = courseDAO.findAll();
        request.setAttribute("courses", courses);

        String applicationContextPath = request.getContextPath();

        if (request.getRequestURI().equals(applicationContextPath + "/"))
        {

            if(security.authorization().getGroupId() != 0)
            request.getRequestDispatcher("home.jsp").forward(request, response);

            else
            request.getRequestDispatcher("enter.jsp").forward(request, response);
        }

        if(security.authorization().getGroupId() != 0)
        {
            request.getRequestDispatcher("home.jsp").forward(request, response);
        }

        request.getRequestDispatcher("login.jsp").forward(request, response);


    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the
left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
```

```
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

Рисунок 3.5 – Реализация класса LoginServlet

Для выхода из сессии используется сервлет LogoutServlet.

```
public class LogoutServlet extends HttpServlet {


    @EJB
    AuthentificationBean security;


    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse re-
sponse)
            throws ServletException, IOException {


        security.logout();
```

```
            request.setAttribute("currentGroup", security.authorization().getGroupId());

            if(security.authorization().getGroupId() == 0)
            request.getRequestDispatcher("enter.jsp").forward(request, response);
        }

        // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
        /**
         * Handles the HTTP <code>GET</code> method.
         *
         * @param request servlet request
         * @param response servlet response
         * @throws ServletException if a servlet-specific error occurs
         * @throws IOException if an I/O error occurs
         */
        @Override
        protected void doGet(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
            processRequest(request, response);
        }

        /**
         * Handles the HTTP <code>POST</code> method.
         *
         * @param request servlet request
         * @param response servlet response
         * @throws ServletException if a servlet-specific error occurs
         * @throws IOException if an I/O error occurs
         */
        @Override
        protected void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
            processRequest(request, response);
        }
```

```
    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

Рисунок 3.6 – реализация класса LogoutServlet

2. Реализация добавления курса.

Операция добавления курса обеспечивается интерфейсом CourseDAO.

Данная функция вызывается путем передачи значения “insertCourse” из скрытого поля с именем “action” в главный контроллер MainController.

```
<form action = "${pageContext.request.contextPath}/MainController">
        <div class="form-group">
            <input type="hidden" name = action value="insertCourse">
            <input type="submit" value="Add course">
        </div>
</form>
```

Рисунок 3.7 – Форма добавления курса

Главный контроллер в свою очередь получает значение данного параметра и выполняет соответствующий фрагмент кода, представленный на рисунке 3.8.

В данном фрагменте кода принимаются значения полей и производится операция вставки курса.

```
if (action.equalsIgnoreCase("insertCourse"))
    {
        String courseId = request.getParameter("courseId");
```

```
  String fullname = request.getParameter("fullname");
  String shortname = request.getParameter("shortname");
  String category = request.getParameter("category");

  if (StringUtils.isNotEmpty(courseId)) {
    Course course = courseDAO.findById(Long.valueOf(courseId));

    if (course.getCourseId() == 0) {
       Course newCourse = new Course();
       newCourse.setCourseId(Long.valueOf(courseId));
       newCourse.setFullname(fullname);
       newCourse.setShortname(shortname);
       newCourse.setCategory(category);

       courseDAO.check(newCourse);



       request.setAttribute("courses", courses);
       request.getRequestDispatcher("courses.jsp").forward(request, response);
    }
    else {
    request.setAttribute("message", "course with id=" + courseId + " exist");
    request.getRequestDispatcher("error.jsp").forward(request, response);
    }
 }
 else
 {

  request.setAttribute("courses", courses);
  request.getRequestDispatcher("addCourse.jsp").forward(request, response);
 }
}
```

Рисунок 3.8 - Реализация функции добавления курса

3. Интеграция редактора кода в систему.

Для этой цели был использован “облачный” редактор Jslinker с открытым исходным кодом.

Данный редактор кода позволяет вводить текст программы и отображать результат компиляции.

В процессе прохождения курса обучаемый будет проходить последовательность закланий используя встроенный в систему редактор кода. В случае получения неверного результата после компиляции отображается соответствующее сообщение, а в случае получения верного результата обучаемому предлагается перейти к следующему заданию.

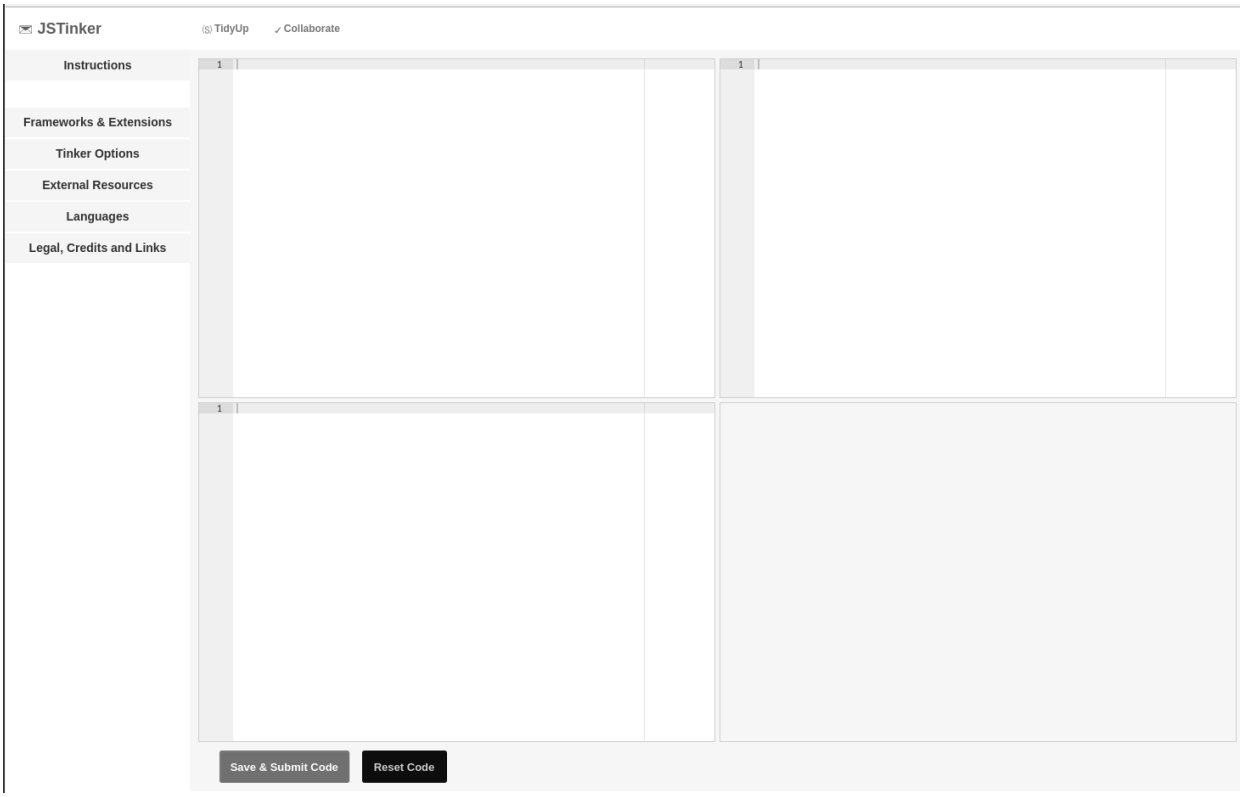

Рисунок 3.9 – Редактор кода встроенный в систему

4. Реализация интерактивного обучения с использованием встроенного в систему редактора кода.

Для данной задачи необходима проверка вводимого пользователем кода и механизм всплывающих подсказок при нажатии на кнопку запуска компиляции кода в случае неверного результата, а также вывод сообщения об успешно выполненном задании с переходом на следующее задание.

5. Реализация добавления заданий различного типа.

Для этой цели был создан модуль EntityModuleDAO позволяющий добавлять задания в базу данных для конкретного курса, производить редактирование и удаление из базы данных.

Для вывода типов заданий, которые может выбрать пользователь был использован текстовой формат обмена данными JSON основанный на JavaScript [12, 123].

Чтение данных осуществляется из файла tasks.json.

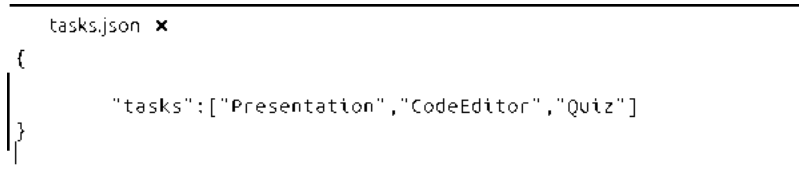

Рисунок 3.10 – Файл tasks.json

Чтение данных обеспечивается интерфейсом LocalEntityModuleDAO.

```
public class LocalEntityModuleDAO implements EntityModuleDAO {
   private Connection connection;
      public LocalEntityModuleDAO() {
      connection = Database.getConnection();
   }

   @Override
     public List<EntityModule> findAll() {

     List<EntityModule> entityModules = new ArrayList<EntityModule>();

     JSONParser parser = new JSONParser();

        try {

                Object obj = parser.parse(new FileReader("c:\\tasks.json"));
```

```
                JSONObject jsonObject = (JSONObject) obj;

                JSONArray msg = (JSONArray) jsonObject.get("tasks");
                Iterator<String> iterator = msg.iterator();
                while (iterator.hasNext()) {
                        EntityModule entityModule = new EntityModule();
                entityModule.setCategory(iterator.next());
                entityModules.add(entityModule);
                }

        } catch (FileNotFoundException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        } catch (ParseException e) {
                e.printStackTrace();
        }

    return entityModules;
   }

}
```

Рисунок 3.11 – Реализация интерфейса интерфейсом LocalEntityModuleDAO

В результате была реализована функция добавления курса, различных учебных материалов и заданий в виде тестов для данного курса. Результат добавления сущностей представлен на рисунке 3.12.

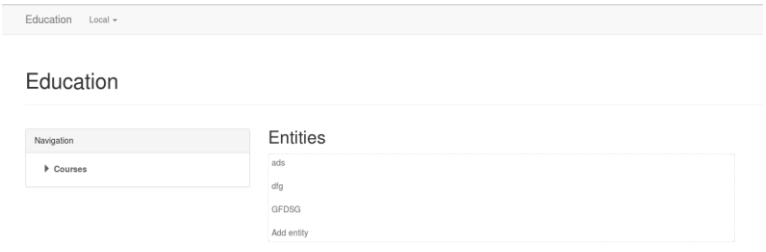

Рисунок 3.12 – Результат добавления заданий

По нажатию на кнопку “Add entity” отображается модальное окно с выбором типа задания.

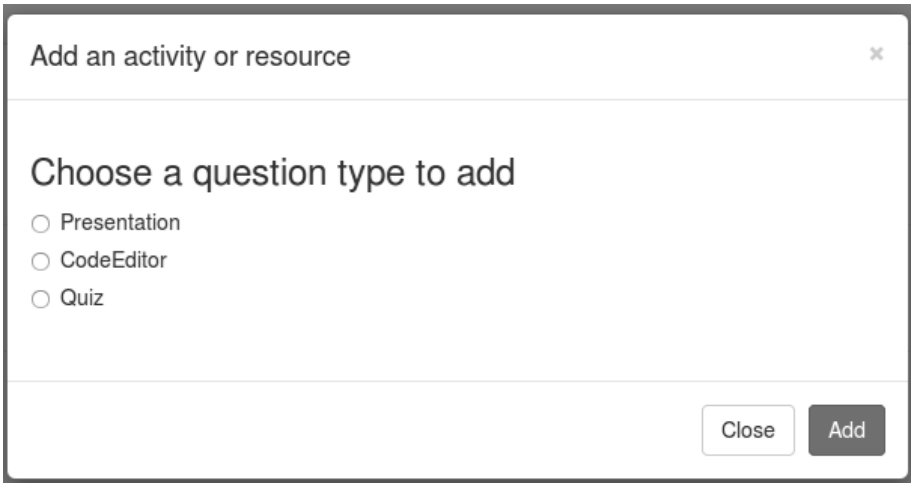

Рисунок 3.13 – Модальное окно добавления заданий для курса

Таким образом, реализована возможность добавления теоретического и практического материала для созданного курса.

Вывод по третьей главер

Была разработана обучающая система, в которой были реализованы механизмы защиты системы, а так же возможность добавлять, удалять и редактировать созданный курс и все задания, входящие в данный курс пользователем, авторизовавшимся как преподаватель.

Обучающий процесс представлен в виде интерактивного обучения, в котором обучающий являясь участником обучающего процесса, изучает теоретический материал, после чего выполняет задания в редакторе кода интегрированного в систему и выполняет тесты по изученному материалу.

Заключение

В процессе анализа существующей системы образования были выявлены процессы, нуждающиеся в автоматизации в частности проблемы связанные с пониманием принципов программирования и их применений на практике вызванные недостаточной интерактивностью процесса обучения разработке прикладных программ.

Исследование предметной области проводилось на основе концептуальной модели по методологиям IDEF0 и DFD и UML с использованием CASE-технологий SADT-методологии моделирования, которые позволили упростить процесс проектирования концептуальной модели.

Анализ функциональной модели “КАК ЕСТЬ” выявил основные недостатки существующей системы обучения связанные с необходимостью обеспечения учеников различным материалом (лекции, пособия), отсутствием автоматизированной проверки выполненных заданий, отсутствие системы самоконтроля своих знаний учеником.

Анализ модели структурного анализа “КАК ДОЛЖНО БЫТЬ” позволил сформулировать основные требования, предъявляемые к системе. На основе, которых был сделан вывод о том, что внедрение автоматизированной обучающей информационной системы повысит эффективность работы всех подразделений учебного заведения и деятельности сторон, вовлеченных в образовательный процесс.

На основе диаграммы вариантов использования были составлены общие требования к функциональному поведению проектируемой системы, разработана исходная концептуальная модель системы для ее последующей детализации в форме логической модели.

На основе диаграммы классов была представлена архитектура автоматизированной обучающей информационной системы.

Благодаря диаграмме развертывания было определено местоположение компонентов на узлах системы на уровне клиента, сервера баз данных и СУБД.

Была разработана обучающая система автоматизирующая процесс обучения разработке программных приложений позволяющая в простой и доступной форме изложить учебный материал по курсу “Разработка программных приложений” и нацелена на выработку навыков программирования с использованием интерактивных элементов обучения.

Список используемой литературы

1. Гонсалес, Э. Изучаем Java EE 7. /Э. Гонсалес. Изучаем Java EE 7 - Питер, 2016. – 640 с.

2. Гупта А. Java EE 7. Основы. / А. Гупта – Вильямс, 2014. – 336 с.

3. Реинжиниринг бизнес-процессов : учеб. пособие для вузов / А.О. Блинов [и др.]; под ред. А.О. Блинова. – Гриф УМО. – М. : ЮНИТИ-ДАНА, 2010. - 341 с.

4. Дэнни Гудман, Майкл Моррисон. JavaScript. Библия пользователя. - СПб: Вильямс, 2010. - 1184 с.

5. Золотов С.Ю. Проектирование информационных систем [Электронный ресурс]: учеб. пособие / С.Ю. Золотов; Томский гос. ун-т систем управления и радиоэлектроники. – Томск: Эль Контент, 2013.- 86 с.

6. Информационные системы и технологии. Проектирование систем с использованием SADT-методологии. Методические указания. – М., 2013 г.

7. Гайкович В.Ю., Ершов Д.В. Основы безопасности информационных технологий - М.: МИФИ, 2013. – 96 с.

8. Карвин, Б. Программирование баз данных SQL. Типичные ошибки и их устранение / Б. Карвин, пер. с англ. Райтман - М.: Рид Групп, 2012.-338 с.

9. Киселев. А. EJB 3 в действии. / А. Киселев. - ДМК Пресс, 2016 . – 600 с.

10. Мурат, Йенер, Алекс Фидом. Java EE. Паттерны проектирования для профессионалов - Питер, 2016. – 240 с.

11. Джим Арлоу, Айла Нейштадт. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование - Символ-Плюс 2012. - 624 с.

12. Никсон, Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScriptи CSSи HTML5-3-е изд. / Р. Никсон - СПБ.: Питер, 2015. - 688 с.

13. Фрэд Лонг, Дхрув Мохиндра, Роберт С. Сиакорд, Дин Сазерленд, Дэвид Свобода. Руководство для программиста на Java. 75 рекомендаций по написанию надежных и защищенных программ - Вильямс, 2014 – 256 с.

14. Хеффельфингер, Д. Java EE 7 и сервер приложений GlassFish 4. / Д, Хеффельфингер – ДМК Пресс, 2016 . - 332 с.

15. Хеффельфингер, Д. Разработка приложений Java EE 7 в NetBens 8. / Д, Хеффельфингер – ДМК Пресс, 2016 . – 348 с.

16.Andy Overton - Введение Connection Pools в Glassfish [Электронный ресурс] // C2B2 [Официальный веб-сайт]: http://blog.c2b2.co.uk/2014/02/an-introduction-to-connection-pools-in.html

17. Justin Ferriman - Какова роль вашей СУО? [Электронный ресурс] // LearnDash [Официальный веб-сайт]: http://www.learndash.com/what-is-the-role-of-your-lms/Justin Ferriman.

18. Justin Ferriman - Предостережение, связанное с СУО индустрией [Электронный ресурс] // LearnDash [Официальный веб-сайт]: http://www.learndash.com/a-caution-about-the-lms-industry

19. Jim Knutson - Выработка долгосрочной стратегии для использования технологии Java EE [Электронный ресурс] // IBM developerWorks [Официальный веб-сайт]: http://www.ibm.com/developerworks/websphere/techjournal/0805_knutson/0805_knutson.html.

20. NetBeans IDE Features - Учебный курс по электронной коммерции в NetBeans. Подготовка представлений страниц и сервлета Controller [Электронный ресурс] // NetBeans [Официальный веб-сайт]: https://netbeans.org/features/index_ru.html.

21.David Salter, Rhawi Dantas. NetBeans IDE 8 Cookbook. - PACKT Publishing, 2014.

22. Gail Anderson, Paul Anderson. JavaFX Rich Client Programming on the NetBeans Platform - Addison-Wesley Professional, 2014.

23. John Brock, Arun Gupta, Geertjan Wielenga. Java EE and HTML5 Enterprise Application Development. - 1st Edition, Oracle Press, 2014.

24. Mario Fusco, Alan Mycroft. Java 8 in Action: Lambdas, Streams, and functional-style programming. - 1st Edition, Manning Publications, 2014

25. Robert Liguori, Patricia Liguori. Java 8 Pocket Guide – O'Reilly Media, 2014.

Приложение

Подключение к базе данных mydb

```
public class Database {
     public static Connection getConnection() {
        try  {
           Class.forName("com.mysql.jdbc.Driver");
           Connection con = DriverManager.getConnection
                 ("jdbc:mysql://localhost:3306/mydb",
                 "admin","admin");
           return con;
        }
        catch(Exception ex) {
           System.out.println("Database.getConnection() Error -->" + ex.getMessage());
           return null;
        }
     }

      public static void close(Connection con) {
        try  {
           con.close();
        }
        catch(Exception ex) {
        }
     }
}
```

Файл LocalCourseDAO.java

```
public class LocalCourseDAO implements CourseDAO {

   private Connection connection;

      public LocalCourseDAO() {
      connection = Database.getConnection();
   }

  /**
    * Anonymous course
    */



   public void check(Course course) {
      try {
         PreparedStatement ps = connection.prepareStatement("select courseId from course where
courseId = ?");
          ps.setLong(1, course.getCourseId());
          ResultSet rs = ps.executeQuery();
          if (rs.next()) // found
          {
```

```
            update(course);
         } else {
            insert(course);
         }
      } catch (Exception ex) {
         System.out.println("Error in check() -->" + ex.getMessage());
      }
   }
   @Override
   public void insert(Course course) {
      try {
         PreparedStatement preparedStatement = connection.prepareStatement("insert into
course(courseId, fullname, shortname, category) values (?, ?, ?, ?)");
         // Parameters start with 1
         preparedStatement.setLong(1, course.getCourseId());
         preparedStatement.setString(2, course.getFullname());
         preparedStatement.setString(3, course.getShortname());
         preparedStatement.setString(4, course.getCategory());
         preparedStatement.executeUpdate();

      } catch (SQLException e) {
         e.printStackTrace();
      }
   }

   @Override
   public void delete(long courseId) {
      try {
         PreparedStatement preparedStatement = connection.prepareStatement("delete from course
where courseId=?");
         // Parameters start with 1
         preparedStatement.setLong(1, courseId);
         preparedStatement.executeUpdate();

      } catch (SQLException e) {
         e.printStackTrace();
      }
   }

   @Override
   public void update(Course course) {
      try {
         PreparedStatement preparedStatement = connection.prepareStatement("update course set
fullname=?, shortname=?, category=?"
               + "where courseId=?");
         // Parameters start with 1

         preparedStatement.setString(1, course.getFullname());
         preparedStatement.setString(2, course.getShortname());
          preparedStatement.setString(3, course.getCategory());
         preparedStatement.setLong(4, course.getCourseId());
         preparedStatement.executeUpdate();
```

```
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public List<Course> findAll() {
        List<Course> courses = new ArrayList<Course>();
        try {
            Statement statement = connection.createStatement();
            ResultSet rs = statement.executeQuery("select * from course");
            while (rs.next()) {
                Course course = new Course();
                course.setCourseId(rs.getLong("courseId"));
                course.setFullname(rs.getString("fullname"));
                course.setShortname(rs.getString("shortname"));
                course.setCategory(rs.getString("category"));
                courses.add(course);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return courses;
    }

    public Course findById(long courseId) {
        Course course = new Course();
        try {
            PreparedStatement preparedStatement = connection.prepareStatement("select * from course
where courseId=?");
            preparedStatement.setLong(1, courseId);
            ResultSet rs = preparedStatement.executeQuery();

            if (rs.next()) {
                course.setCourseId(rs.getLong("courseId"));
                course.setFullname(rs.getString("fullname"));
                course.setShortname(rs.getString("shortname"));
                course.setCategory(rs.getString("category"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return course;
    }
```

Файл MainController.java

```
public class MainController extends HttpServlet {
```

```
    CourseDAO courseDAO = DAOFacto-
ry.getDAOFactory(DAOFactory.DAOType.LOCAL).getCourseDAO();
    EntityDAO entityQuizDAO = DAOFacto-
ry.getDAOFactory(DAOFactory.DAOType.LOCAL).getQuizDAO();
    EntityDAO entityCodeEditorDAO = DAOFacto-
ry.getDAOFactory(DAOFactory.DAOType.LOCAL).getCodeEditorDAO();
    EntityModuleDAO entityModuleDAO = DAOFacto-
ry.getDAOFactory(DAOFactory.DAOType.LOCAL).getEntityModuleDAO();
    ExerciseDAO exerciseDAO = DAOFacto-
ry.getDAOFactory(DAOFactory.DAOType.LOCAL).getExerciseDAO();

    @EJB
    AuthentificationBean security;
    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {


        String action = request.getParameter("action");

        List<Course> courses = courseDAO.findAll();
        request.setAttribute("courses", courses);

        String applicationContextPath = request.getContextPath();

        if (request.getRequestURI().equals(applicationContextPath + "/"))
        {


            try {
                security.login();
            } catch (ru.ncedu.core.security.SecurityException ex) {
                request.setAttribute("loginStatusText", ex.getMessage());
            }

            request.setAttribute("currentGroup", security.authorization().getGroupId());

//            if(security.authorization().getGroupId() != 0)
            request.getRequestDispatcher("home.jsp").forward(request, response);

//            else
//            request.getRequestDispatcher("enter.jsp").forward(request, response);
```

```
}

if (action.equalsIgnoreCase("login"))
{
   try {
   security.login();
   } catch (ru.ncedu.core.security.SecurityException ex) {
      request.setAttribute("loginStatusText", ex.getMessage());
   }

   request.setAttribute("currentGroup", security.authorization().getGroupId());

   if(security.authorization().getGroupId() != 0)
   {
      request.getRequestDispatcher("home.jsp").forward(request, response);
   }

   request.getRequestDispatcher("login.jsp").forward(request, response);
}


if (action.equalsIgnoreCase("logout"))
{
   security.logout();

   request.setAttribute("currentGroup", security.authorization().getGroupId());

   request.getRequestDispatcher("home.jsp").forward(request, response);
}


if (action.equalsIgnoreCase("coursePage"))
{

   String courseId = request.getParameter("courseId");


   if (courseId == null ) {


   String id = request.getParameter("id");
   request.setAttribute("id", id);


   List <Entity> quizes = entityQuizDAO.findByAllById(Long.valueOf(id));
   request.setAttribute("quizes", quizes);

   List <Entity> codeEditors = entityCodeEditorDAO.findByAllById(Long.valueOf(id));
   request.setAttribute("codeEditors", codeEditors);

   List <EntityModule> entityModules = entityModuleDAO.findAll();
   request.setAttribute("entityModules", entityModules);
```

```
    request.getRequestDispatcher("coursePage.jsp").forward(request, response);

    }

}

if (action.equalsIgnoreCase("quiz"))
{

    String id = request.getParameter("id");
    request.setAttribute("id", id);

    String courseId = request.getParameter("courseId");
    request.setAttribute("courseId", courseId);

    Entity entity = entityQuizDAO.findById(Long.valueOf(id));
    request.setAttribute("entity", entity);

    request.getRequestDispatcher("quiz.jsp").forward(request, response);
}

if (action.equalsIgnoreCase("codeEditor"))
{


    String id = request.getParameter("id");
    String courseId = request.getParameter("courseId");
    request.setAttribute("courseId", courseId);

    List <Exercise> exercises = exerciseDAO.findAllById(Long.valueOf(id));
    request.setAttribute("exercises", exercises);
    ;

    Entity entity = entityCodeEditorDAO.findById(Long.valueOf(id));
    request.setAttribute("entity", entity);
    request.getRequestDispatcher("codeEditor.jsp").forward(request, response);
}


if (action.equalsIgnoreCase("editEntity"))
{

    String id = request.getParameter("id");

    Entity entity = entityQuizDAO.findById(Long.valueOf(id));
    request.setAttribute("entity", entity);
    request.getRequestDispatcher("editEntity.jsp").forward(request, response);
}

if (action.equalsIgnoreCase("insertCourse"))
{
    String courseId = request.getParameter("courseId");
```

```
        String fullname = request.getParameter("fullname");
        String shortname = request.getParameter("shortname");
        String category = request.getParameter("category");

        if (StringUtils.isNotEmpty(courseId)) {
          Course course = courseDAO.findById(Long.valueOf(courseId));

          if (course.getCourseId() == 0) {
             Course newCourse = new Course();
             newCourse.setCourseId(Long.valueOf(courseId));
             newCourse.setFullname(fullname);
             newCourse.setShortname(shortname);
             newCourse.setCategory(category);

             courseDAO.check(newCourse);


             request.setAttribute("courses", courses);
             request.getRequestDispatcher("courses.jsp").forward(request, response);
          }
          else {
          request.setAttribute("message", "course with id=" + courseId + " exist");
          request.getRequestDispatcher("error.jsp").forward(request, response);
          }
        }
        else
        {

          request.setAttribute("courses", courses);
          request.getRequestDispatcher("addCourse.jsp").forward(request, response);
        }
      }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the
left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
```

```
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```