РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра информационных технологий

УТВЕРЖДАЮ

Заведующий кафедрой
информационных технологий
к.ф.-м.н
_____ Ю.Н. Орлов

«____»_____ 20__ г.

КУРСОВАЯ РАБОТА

на тему

«Разработка модели информационной системы интернет-портала онлайн-образования на основе UML»

по дисциплине «Моделирование сложно структурированных систем»

Выполнил
Студент группы НПИбд-01-18
Студенческий билет
№:1032162043
Еременко Артем Геннадьевич

«___» _____ 20__ г.

Научный руководитель
Доцент кафедры
Информационных технологий,
к.ф.-м.н., И. В. Смирнов

Научный консультант
Доцент кафедры
Информационных технологий,
к.ф.-м.н., М. В. Хачумов

Москва 2021

Содержание

Аннотация

В данной курсовой работе описывается техническое задание по разработке модели информационной системы интернет-портала онлайн-образования на основе UML. Также проводится обзор на язык UML: изучается то, какие строительные блоки в нём присутствуют, и как их следует применять; анализируются общие механизмы действия. Далее разрабатывается выполнение практической части в виде построения одиннадцати диаграмм по модели информационной системы интернет-портала онлайн-образования: 4 диаграммы использования, 4 диаграммы классов и 3 диаграммы последовательности действий для отдельно взятых вариантов использования.

Введение

UML (Unified Modeling Language) – это унифицированный графический язык моделирования для описания, визуализации, проектирования и документирования ОО (объектно-ориентированных) систем [1]. UML призван поддерживать процесс моделирования ПС (программных систем) на основе ОО подхода, организовывать взаимосвязь концептуальных и программных понятий, отражать проблемы масштабирования сложных систем. Модели на UML используются на всех этапах жизненного цикла ПС, начиная с бизнес-анализа и заканчивая сопровождением системы. Разные организации могут применять UML по своему усмотрению в зависимости от своих проблемных областей и используемых технологий.

К середине 90-х годов различными авторами было предложено несколько десятков методов ОО моделирования, каждый из которых использовал свою графическую нотацию. При этом любой их этих методов имел свои сильные стороны, но не позволял построить достаточно полную модель ПС, показать ее «со всех сторон», то есть, все необходимые проекции. К тому же отсутствие стандарта ОО моделирования затрудняло для разработчиков выбор наиболее подходящего метода, что препятствовало широкому распространению ОО подхода к разработке ПС.

По запросу консорциума OMG (Object Management Group) [2] – организации, ответственной за принятие стандартов в области объектных технологий и баз

данных назревшая проблема унификации и стандартизации была решена авторами трех наиболее популярных ОО методов - Г.Бучем, Д.Рамбо и А.Джекобсоном [3], которые объединенными усилиями создали версию UML 1.1 [4], утвержденную OMG в 1997 году в качестве стандарта.

Постановка задачи и методы разработки информационной системы

1 Постановка задачи

Составить техническое задание.

Сделать обзор UML.

Разработать модель информационной системы интернет-портала онлайн-образования на основе UML.

2 Техническое задание

Информационная система предназначена для прохождения онлайн-занятий по освоению выбранного курса обучения.

Пользователи системы:

1. Неавторизованные посетители сайта (Посетитель)
2. Авторизованные студенты (Студент)
3. Преподаватели курсов (Преподаватель)
4. Администратор сайта (Администратор)

Система должна предоставлять посетителю следующие возможности:

1) Просмотреть информацию о курсах
2) Просмотреть информацию об организации
3) Обратиться в службу поддержки
4) Авторизоваться
   a) Войти в аккаунт
   b) Создать аккаунт

Система должна предоставлять студенту следующие возможности:

1) Купить курс

a) Выбрать метод оплаты
   i) Выбрать оплату картой онлайн
   ii) Выбрать оплату платёжным сервисом
   iii) Оформить рассрочку на покупку
b) Применить промо-код на скидку

2) Открыть доступный курс
   a) Перейти в модуль
      i) Открыть урок
         (1) Просмотреть обучающие видео урока
         (2) Сдать домашнюю работу
         (3) Написать курирующему преподавателю

3) Просмотреть свой профиль
   a) Редактировать личные данные
   b) Изменить пароль

Система должна предоставлять администратору следующие возможности:

1) Управлять курсами
   a) Добавить курс
   b) Удалить курс
   c) Изменить курс
      i) Изменить название курса
      ii) Добавить модуль
      iii) Удалить модуль
      iv) Изменить модуль
         (1) Изменить название модуля
         (2) Добавить урок
         (3) Удалить урок
         (4) Изменить урок
            (a) Добавить видео
            (b) Удалить видео
            (c) Добавить домашнюю работу
            (d) Удалить домашнюю работу

2) Управлять пользователями
   a) Добавить пользователя
   b) Удалить пользователя
   c) Редактировать информацию пользователя
3) Прочитать обращения из службы поддержки

Система должна предоставлять преподавателю следующие возможности:

1) Проверить домашнее задание
   a) Вернуть домашнее задание
   b) Принять домашнее задание
   c) Оставить комментарий
2) Просмотреть свой профиль
   a) Редактировать личные данные
   b) Изменить пароль

3 Обзор и исследование языка моделирования UML для разработки информационных систем

3.1 Строительные блоки UML

Словарь языка UML включает три вида строительных блоков:

- сущности;
- отношения;
- диаграммы.

3.1.1 Сущности

Сущности – это абстракции, являющиеся основными элементами модели. Они являются наиболее важными строительными блоками UML. В UML имеется четыре типа сущностей:

- структурные;
- поведенческие;
- группирующие;
- аннотационные.

Структурные сущности

Структурные сущности определяют статическую часть модели. Они представляют физические и концептуальные элементы. Ниже приведены краткие описания структурных сущностей.

Класс — это описание совокупности объектов с общими атрибутами, операциями, отношениями и семантикой. Класс реализует один или несколько интерфейсов. Графически класс изображается в виде прямоугольника, в котором обычно записаны его имя, атрибуты и операции, как показано на рис. 1.

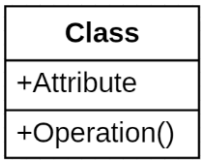

Рис. 1. Класс

Интерфейс — это совокупность операций, которые определяют сервис (набор услуг), предоставляемый классом или компонентом. Таким образом, интерфейс описывает видимое извне поведение элемента. Интерфейс может представлять поведение класса или компонента полностью или частично; он определяет только спецификации операций (сигнатуры), но никогда - их реализации. Графически интерфейс изображается в виде круга, под которым пишется его имя, как показано на рис. 2. Интерфейс редко существует сам по себе - обычно он присоединяется к реализующему его классу или компоненту.

Рис. 2. Интерфейс

Кооперация — определяет взаимодействие; она представляет собой совокупность ролей и других элементов, которые, работая совместно, производят некоторый кооперативный эффект, не сводящийся к простой сумме слагаемых. Кооперация, следовательно, имеет как структурный, так и поведенческий аспект. Один и тот же класс может принимать участие в нескольких кооперациях; таким образом, они являются реализацией образцов поведения, формирующих систему.

Графически кооперация изображается в виде эллипса, ограниченного пунктирной линией, в который обычно заключено только имя, как показано на рис. 3.

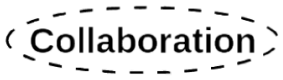

Рис. 3. Кооперация

Вариант использования — это описание последовательности выполняемых системой действий, которая производит наблюдаемый результат, значимый для какого-то определенного актера (Actor). Прецедент применяется для структурирования поведенческих сущностей модели. Прецеденты реализуются посредством кооперации. Графически прецедент изображается в виде ограниченного непрерывной линией эллипса, обычно содержащего только его имя как показано на рис. 4.

Рис. 4. Вариант использования

Компонент — это физическая заменяемая часть системы, которая соответствует некоторому набору интерфейсов и обеспечивает его реализацию. В системе можно встретить различные виды устанавливаемых компонентов, а также компоненты, являющиеся артефактами процесса разработки, например файлы исходного кода. Компонент, как правило, представляет собой физическую упаковку логических элементов, таких как классы, интерфейсы и кооперации. Графически компонент изображается в виде прямоугольника с вкладками, содержащего обычно только имя, как показано на рис. 5.

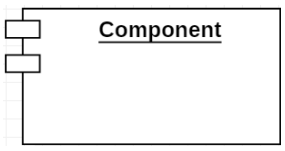

Рис. 5. Компонент

Узел — это элемент реальной (физической) системы, который существует во время функционирования программного комплекса и представляет собой вычислительный ресурс, обычно обладающий как минимум некоторым объемом памяти, а часто еще и способностью обработки. Совокупность компонентов может размещаться в узле, а также мигрировать с одного узла на другой. Графически узел изображается в виде куба, обычно содержащего только имя, как показано на рис. 6.

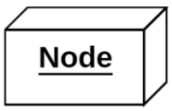

Рис. 6. Узел

Поведенческие сущности

Поведенческие сущности состоят из динамических частей моделей UML. Это глаголы языка: они описывают поведение модели во времени и пространстве. Существует всего два основных типа поведенческих сущностей.

Взаимодействие — это поведение, суть которого заключается в обмене сообщениями (Messages) между объектами в рамках конкретного контекста для достижения определенной цели. С помощью взаимодействия можно описать как отдельную операцию, так и поведение совокупности объектов. Взаимодействие предполагает ряд других элементов, таких как сообщения, последовательности действий (поведение, инициированное сообщением) и связи (между объектами). Графически сообщения изображаются в виде стрелки, над которой почти всегда пишется имя соответствующей операции, как показано на рис. 7.

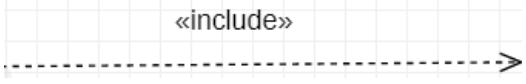

Рис. 7. Взаимодействие

Состояние — это некое положение в жизни объекта, при котором он удовлетворяет определенному условию, выполняет некоторое действие или

ожидает события. Состояние объекта можно описать с помощью значений одного или нескольких атрибутов класса [5]. Графически состояние изображается в виде прямоугольника с закругленными углами, содержащего имя и, возможно, подсостояния (см. рис. 8).

Рис. 8. Состояние

Группирующие сущности

Группирующие сущности являются организующими частями модели UML. Это блоки, на которые можно разложить модель. Есть только одна первичная группирующая сущность, а именно пакет.

Пакеты — представляют собой универсальный механизм организации элементов в группы. В пакет можно поместить структурные, поведенческие и даже другие группирующие сущности. [6] В отличие от компонентов, существующих во время работы программы, пакеты носят чисто концептуальный характер, то есть существуют только во время разработки. Изображается пакет в виде папки с закладкой, содержащей, как правило, только имя и иногда - содержимое (см. рис. 9).

Рис. 9. Пакет

Аннотационные сущности

Аннотационные сущности могут быть определены как механизм для сбора замечаний, описаний и комментариев элементов модели UML. Имеется только один базовый тип аннотационных элементов – примечание.

Примечание — это просто символ для изображения комментариев или ограничений, присоединенных к элементу или группе элементов. Графически

примечание изображается в виде прямоугольника с загнутым краем, содержащим текстовый или графический комментарий, как показано на рис. 10.

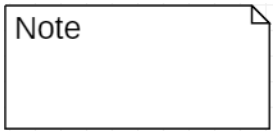

Рис. 10. Примечание

3.1.2 Отношения

Отношения — это еще один важнейший строительный блок UML. Он показывает, как элементы связаны друг с другом, и эта связь описывает функциональность приложения.

Есть четыре вида доступных отношений:

- зависимость;
- ассоциация;
- обобщение;
- реализация.

Зависимость

Зависимость — это семантическое отношение между двумя сущностями, при котором изменение одной из них, независимой, может повлиять на семантику другой, зависимой. Графически зависимость изображается в виде прямой пунктирной линии, часто со стрелкой, которая может содержать метку (см. рис. 11).

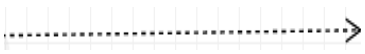

Рис. 11. Зависимость

Ассоциация

Ассоциация — структурное отношение, являющееся одним из фундаментальных понятий в языке UML, которое описывает совокупность связей, где под связью понимается соединение между объектами. Ассоциация может использоваться на различных канонических диаграммах при построении

визуальных моделей [7]. Разновидностью ассоциации является агрегирование (Aggregation) - так называют структурное отношение между целым и его частями. Графически ассоциация изображается в виде прямой линии (иногда завершающейся стрелкой или содержащей метку), рядом с которой могут присутствовать дополнительные обозначения, на пример кратность и имена ролей. На рис. 12 показан пример отношений этого типа.

Рис. 12. Ассоциация

Обобщение

Обобщение — это отношение "специализация/обобщение", при котором объект специализированного элемента (потомок) может быть подставлен вместо объекта обобщенного элемента (родителя или предка). Таким образом, потомок (Child) наследует структуру и поведение своего родителя (Parent). Графически отношение обобщения изображается в виде линии с не закрашенной стрелкой, указывающей на родителя, как показано на рис. 13.

Рис. 13. Обобщение

Реализация

Реализация — это семантическое отношение между классификаторами, при котором один классификатор определяет "контракт", а другой гарантирует его выполнение. Отношения реализации встречаются в двух случаях: во-первых, между интерфейсами и реализующими их классами или компонентами, а во-вторых, между прецедентами и реализующими их кооперациями. Отношение реализации изображается в виде пунктирной линии с не закрашенной стрелкой, как нечто среднее между отношениями обобщения и зависимости [8] (см. рис. 14).

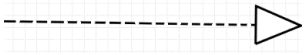

Рис. 14. Реализация

Отношения связывают различные сущности; диаграммы группируют представляющие интерес совокупности сущностей.

3.1.3 Диаграммы

Диаграмма — это графическое представление набора элементов, изображаемое чаще всего в виде связанного графа с вершинами (сущностями) и ребрами (отношениями). Диаграммы рисуют для визуализации системы с разных точек зрения. Диаграмма - в некотором смысле одна из проекций системы. Как правило, за исключением наиболее тривиальных случаев, диаграммы дают свернутое представление элементов, из которых составлена система. Один и тот же элемент может присутствовать во всех диаграммах, или только в нескольких (самый распространенный вариант), или не присутствовать ни в одной (очень редко). Теоретически диаграммы могут содержать любые комбинации сущностей и отношений. На практике, однако, применяется сравнительно небольшое количество типовых комбинаций, соответствующих пяти наиболее употребительным видам, которые составляют архитектуру программной системы. Таким образом, в UML выделяют восемь типов диаграмм, которые разделяются на два глобальных типа:

- Структурные диаграммы;
- Диаграммы поведения.

Структурные диаграммы

- **Классов** — позволяют описать модель классов и их отношения при помощи графической системы обозначений [9]. При моделировании объектно-ориентированных систем этот тип диаграмм используют чаще всего. Диаграммы классов соответствуют статическому виду системы с точки зрения проектирования. Диаграммы классов, которые включают активные классы, соответствуют статическому виду системы с точки зрения процессов.
- **Объектов** — представлены объекты и отношения между ними. Они являются статическими "фотографиями" экземпляров сущностей, показанных на диаграммах классов. Диаграммы объектов, как и диаграммы классов, относятся к статическому виду системы с точки зрения проектирования или процессов, но с расчетом на настоящую или макетную реализацию.

- **Компонентов** — представлена организация совокупности компонентов и существующие между ними зависимости. Диаграммы компонентов относятся к статическому виду системы с точки зрения реализации. Они могут быть соотнесены с диаграммами классов, так как компонент обычно отображается на один или несколько классов, интерфейсов или кооперации.
- **Размещения** — представлена конфигурация обрабатывающих узлов системы и размещенных в них компонентов. Диаграммы развертывания относятся к статическому виду архитектуры системы с точки зрения развертывания. Они связаны с диаграммами компонентов, поскольку в узле обычно размещаются один или несколько компонентов.

Диаграммы поведения

- **Последовательности** — отражают временную упорядоченность сообщений. С помощью них можно проиллюстрировать взаимодействие исполнителя с системой и операции, выполнение которых при этом инициализируется [10]. Являются частным случаем диаграммы взаимодействия, на которой представлены связи между объектами; показаны, в частности, сообщения, которыми объекты могут обмениваться. Могут быть преобразованы в диаграмму коммуникации.
- **Коммуникации** — отражает структурную организацию обменивающихся сообщениями объектов. Также является частным случаем диаграммы взаимодействия. Может быть преобразована в диаграмму последовательности.
- **Автомата / состояний** — на диаграммах состояний представлен автомат, включающий в себя состояния, переходы, события и виды действий. Диаграммы состояний относятся к динамическому виду системы; особенно они важны при моделировании поведения интерфейса, класса или кооперации. Они акцентируют внимание на поведении объекта, зависящем от последовательности событий, что очень полезно для моделирования реактивных систем.

- **Деятельности** — это частный случай диаграммы состояний; на ней представлены переходы потока управления от одной деятельности к другой внутри системы. Диаграммы деятельности относятся к динамическому виду системы; они наиболее важны при моделировании ее функционирования и отражают поток управления между объектами.

3.2 Общие механизмы UML

Общие механизмы UML последовательно применяются по всему языку моделирования. Всего выделяют четыре общих механизма:

- спецификации (описание «заднего плана» модели);
- дополнения (возможности дополненного описания любого символа UML);
- принятые деления (описывают конкретные способы представления объектов реального мира в модели);
- механизмы расширения (применяются в случае, когда базовые возможности UML не удовлетворяют выдвигаемым требованиям).

Спецификации — это текстовое описание модели, которое отражает ее суть. Любая модель может быть представлена как графически, так и в текстовом виде — в виде спецификаций. Спецификации также используются для удобства моделирования, поскольку ряд элементов модели может присутствовать в спецификации и отсутствовать на диаграмме.

UML не сводится к диаграммам. Напротив, диаграммы служат для визуального отражения модели интеллектуальной системы. Соответственно, одних только графических диаграмм недостаточно для описания модели. Любая диаграмма должна содержать спецификации, благодаря которым она и оказывается вплетенной в общую модель. Только диаграмма, обладающая исчерпывающей спецификацией, может рассматриваться как по-настоящему полезная.

Дополнения используются в UML для того, чтобы представить какой- либо элемент диаграмм с необходимой полнотой. Визуализация как таковая не всегда достаточна, и в этом случае востребованными оказываются дополнения.

Дополнения нужны, чтобы отразить на диаграмме важные характеристики модели, которые не представляется возможным показать при помощи визуальных методов (например, из-за перегруженности диаграммы). Как правило, дополнения изображаются в форме прямоугольника, который содержит информацию об элементе модели.

Другой важный общий механизм UML — это принятые деления. Во-первых, следует рассмотреть деление «Классификатор — Экземпляр». В качестве классификатора понимается абстрактное понятие (например, сотрудник предприятия). Соответственно, экземпляром является конкретный объект (например, «экземпляр» маркетолог Иванов Иван Иванович).

Второе принятое деление — это «Интерфейс — Реализация». Это деление позволяет отделить что выполняется, от того, как это выполняется. Например, в случае самолета приборная панель и штурвал будут рассматриваться в качестве интерфейса, тогда как устройство самолета будет реализацией. Как правило, конкретные виды реализации обеспечивают существование того или иного интерфейса, но возможна и обратная ситуация.

И, наконец, следует сказать несколько слов о механизмах расширения. Выделяют три основных механизма:

- ограничения (расширяют семантику элемента, позволяя тем самым добавлять новые правила или изменять старые);
- стереотипы (возможность определять новые элементы модели на основании существующих);
- теговые величины (возможность добавлять новую специальную информацию в спецификации элемента).

4 Разработка диаграмм UML

Как уже описывалось в техническом задании, в системе были выделены четыре вида пользователей: Посетитель, Студент, Преподаватель, Администратор. Для каждого пользователя составлены по одной диаграмме использования (рис. 15-18), по одной диаграмме классов (рис. 19-22) и по одной диаграмме

последовательности действий для наиболее важных вариантов использования (рис. 23-25).

4.1 Диаграммы использования

В представленной системе пользователь Посетитель является по сути родителем пользователей Преподаватель и Студент, передавая им свои варианты использования, однако с той лишь разницей, что Преподаватель лишён возможности самостоятельно зарегистрироваться, и нуждается в создании своего аккаунта со стороны администратора.

Посетитель имеет возможность совершать следующие действия:

1. Просмотреть информацию о курсе – означает, что любой пользователь системы может ознакомиться с информацией об актуальных курсах подготовки: название курса, описание, модули, количество часов лекций, предполагаемое время обучения и цена.
2. Просмотреть информацию об организации – означает, что любой пользователь системы может ознакомиться с информацией о сайте (об организации) на базе которой проходит обучение: название, описание, физический адрес, контакты и юридическая информация.
3. Обратиться в службу поддержки – это возможность любого пользователя, даже незарегистрированного, написать обращение в службу поддержки, при этом оставив адрес своей электронной почты для ответа.
4. Авторизоваться – это возможность для неавторизованного Посетителя войти в систему, перейдя в статус Студента или Преподавателя, или же зарегистрировать аккаунт самостоятельно, тем самым став Студентом.

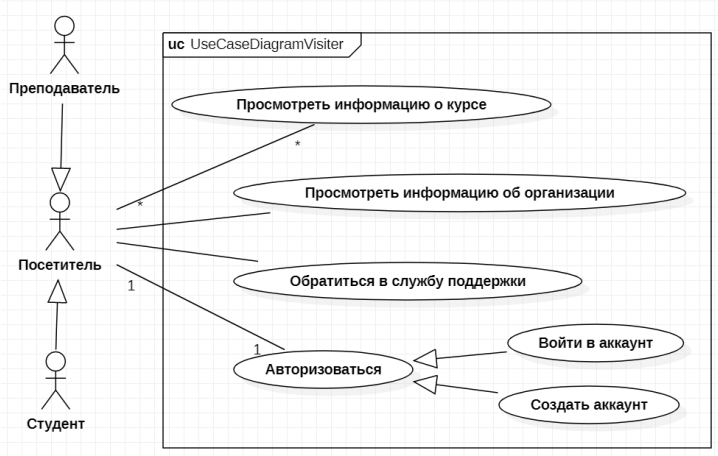

Рис. 15. Диаграмма использования для посетителя

Студент – это дочерний класс Посетителя. Студентом становится Посетитель сайта, который прошёл процесс авторизации.

Студент имеет возможность совершать следующие действия:

1. Купить курс – это приобретение доступа к выбранному курсу путём оплаты такового. Для совершение этого действия следует прежде всего выбрать метод оплаты: банковская карта, платёжный сервис (PayPal, Qiwi, Yandex.Деньги и т.п.) или же оформление рассрочки на покупку через банк партнёра, что по сути является беспроцентным кредитом.
2. Открыть доступный курс – это процесс прохождения уже приобретённого курса или множества курсов. Процесс включает в себя переходы в меньшие единицы, являющиеся составными частями курса (модули, уроки). Каждый урок включает в себя обучающее видео и/или домашнее задание, которое требуется сдавать через систему.
3. Просмотреть свой профиль – это действие подразумевает просмотр данных, предоставленных пользователем при регистрации, а также их изменение (личные данные, пароль).

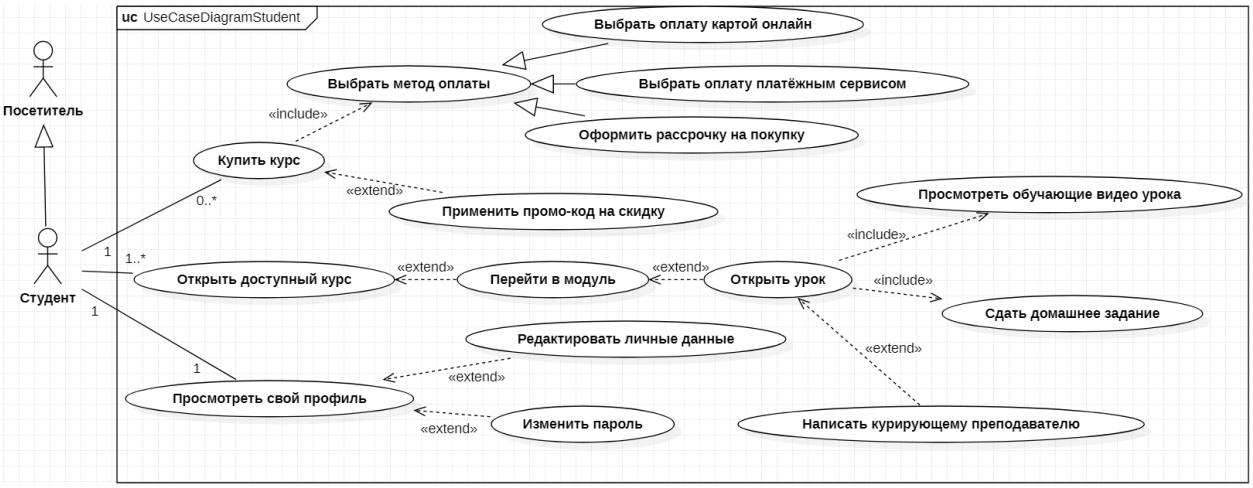

Рис. 16. Диаграмма использования для студента

В представленной диаграмме Администратор является родителем пользователя Преподаватель и по усмотрению может предоставлять ему права на некоторые действия, таки как управление курсами, управление пользователями и прочтение обращений в службу поддержки.

Администратор имеет возможность совершать следующие действия:

1. Управлять курсами – это возможность пользователя добавлять новые, а также изменять или удалять существующие курсы. В ходе изменения курса существует возможность изменить его название или состав, добавив, изменив или удалив модуль. Изменяя модуль, Администратор также может изменить его название или состав, добавив, изменив или удалив урок. В ходе редактирования содержимого урока пользователь имеет возможность добавлять и удалять обучающее видео или домашнюю работу.
2. Управлять пользователями – под этим действием подразумевается администрирование зарегистрированными пользователями посредством информационной системы. В ходе данного управления пользователь может быть добавлен или удалён администратором вручную, а также информация о нём изменена, если речь идёт о пользователе Преподаватель.
3. Прочитать обращения из службы поддержки – это действие направлено на проверку сообщений, которые могут поступить, по сути, от любого

пользователя системы. Долг администратора в данном случае заключается в том, чтобы разобраться в ситуации и отправить ответ на ту электронную почту, которая была указана в обращении.

Рис. 17. Диаграмма использования для администратора

Преподаватель является дочерним пользователем Администратора и Посетителя, тем самым унаследовав возможности обоих актёров, за исключением возможности самостоятельно создавать аккаунт.

Преподаватель имеет возможность совершать следующие действия:

1. Проверить домашнее задание – это возможность пользователя просмотреть то, что в качестве домашнего задания ему прислал Студент, и после этого принять решение о том, засчитывается работа или нет. В качестве дополнения, за преподавателем остаётся возможность оставить свои комментарии, тем самым похвалив, направив в нужном направлении, дав совет или ответив на прямой вопрос Студента.
2. Просмотреть свой профиль – это действие подразумевает просмотр данных, введённых Администратором при регистрации пользователя в системе, а также их изменение (личные данные, пароль).

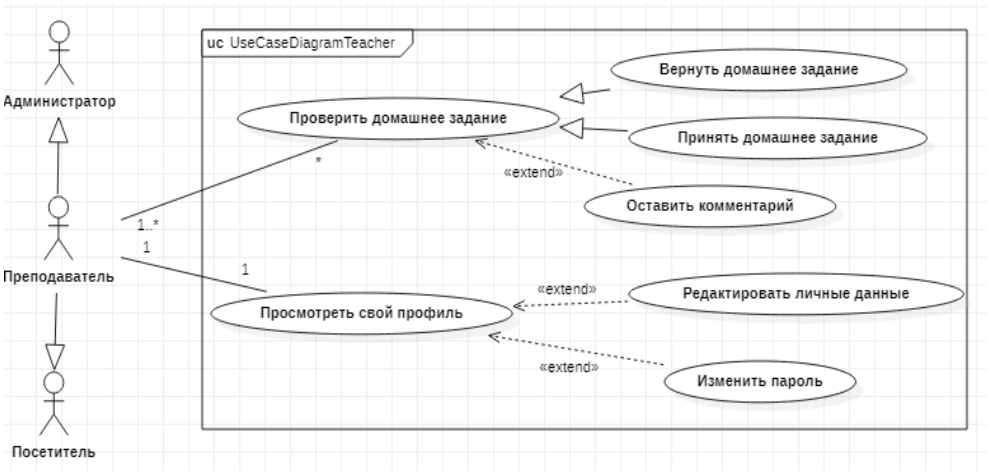

Рис. 18. Диаграмма использования для преподавателя

4.2 Диаграммы классов

На основе диаграмм использования были разработаны диаграммы классов для каждого типа пользователей (рис. 19-22).

IПосетитель представляет собой пользовательский интерфейс, благодаря которому можно просмотреть информацию об имеющихся курсах или об организации (онлайн-портале), авторизоваться в системе, обратиться в службу поддержки, описав возникшую проблему и указав электронную почту для обратной связи.

При авторизации происходит переход к интерфейсу IАвторизация, в котором возможно одно из двух действий: вход в существующий аккаунт с указанием электронной почты и пароля для идентификации, создание аккаунта с введением таких регистрационных данных, как имя, пол, город, день рождения, электронная почта и пароль.

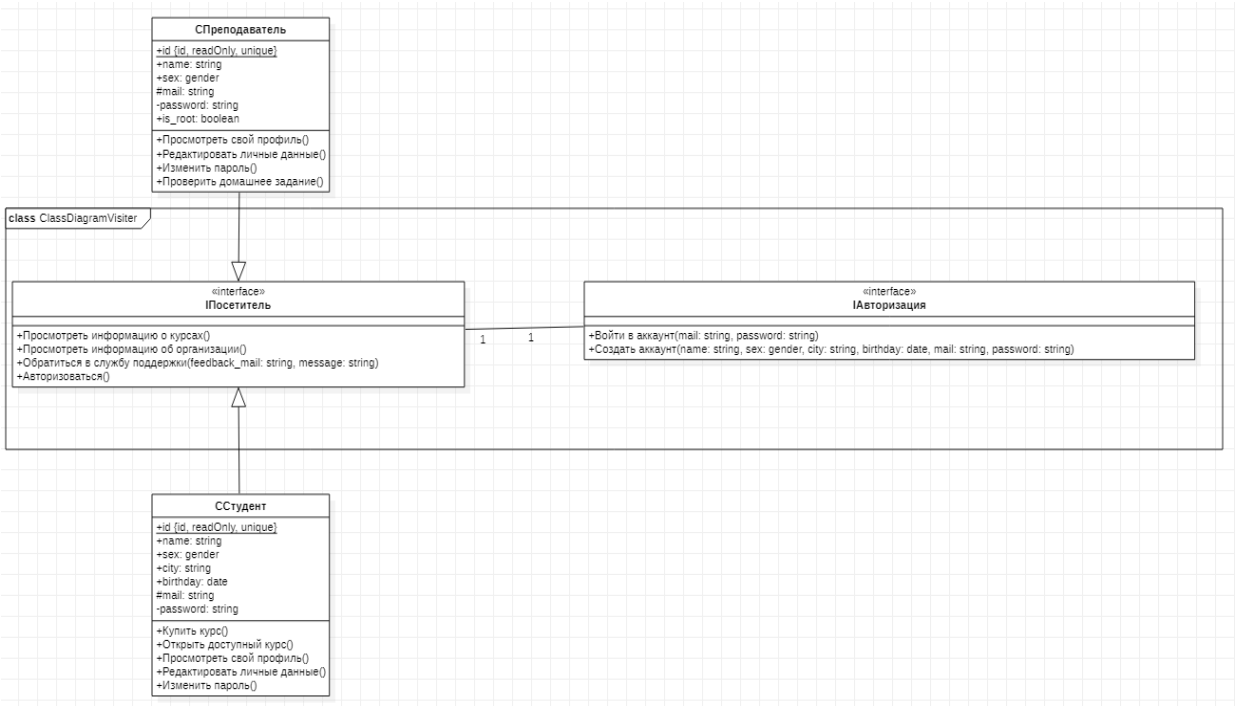

Рис. 19. Диаграмма классов для посетителя

ССтудент представляет собой класс, каждый экземпляр которого имеет следующие атрибуты: уникальный идентификатор id, имя name типа string, пол sex собственного типа gender, город проживания city типа string, день рождения birthday типа date, электронная почта mail типа string, пароль password типа string. Каждый экземпляр класса имеет следующий набор операций: «Купить курс», «Открыть доступный курс», «Просмотреть свой профиль», «Редактировать личные данные», «Изменить пароль».

При вызове операции «Открыть доступный курс» происходит переход к экземпляру класса СКурс, который имеет следующие атрибуты: уникальный идентификатор course_id, название курса title типа string, сфера, к которой относится курс, field типа string. Аналогичный переход происходит при вызове операции «Купить курс», однако с тем отличием, что при покупке курса происходит обращение к интерфейсу IОплата, в котором можно оплатить выбранный курс, выбрать способ оплаты и применить промо-код на скидку. При выборе способа оплаты вызывается соответствующий интерфейс со следующими вариантами оплаты: «Выбрать оплату картой онлайн», «Выбрать оплату платёжным сервисом», «Оформить рассрочку на покупку».

Если же курс уже приобретён, то появляется возможность выполнить операцию «Перейти в модуль», тем самым перейдя к экземпляру класса СМодуль, который имеет следующие атрибуты: номер модуля module_number типа int, название модуля module_title типа string. Каждый экземпляр класса имеет единственную операцию «Открыть урок», при котором осуществляется переход к экземпляру класса СУрок.

СУрок представляет собой класс, каждый экземпляр которого имеет следующие атрибуты: номер урока lesson_number типа int, название урока lesson_title типа string, токен урока lesson_token типа string, атрибут наличия обучающего видео is_video типа boolean, атрибут наличия домашней работы is_homework типа boolean. Каждый экземпляр класса имеет следующий набор операций: «Просмотреть обучающие видео урока», «Сдать домашнюю работу», «Написать курирующему преподавателю».

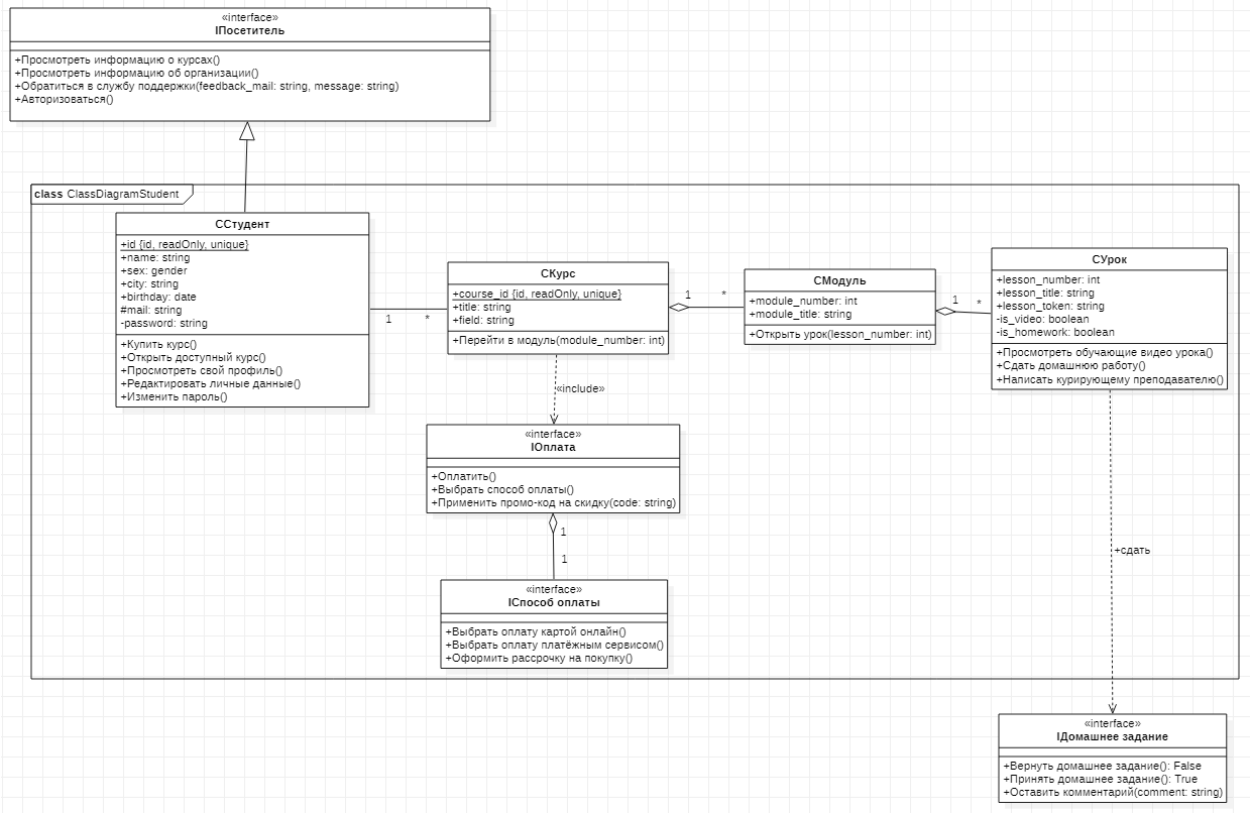

Рис. 20. Диаграмма классов для студента

САдминистратор представляет собой класс, каждый экземпляр которого имеет следующие атрибуты: уникальный идентификатор id, имя name типа string, электронная почта mail типа string, пароль password типа string, константный атрибут

is_root типа boolean равный True. Каждый экземпляр класса имеет следующий набор операций при условии is_root=True, что по сути своей означает наличие прав доступа администратора: «Добавить курс», «Удалить курс», «Изменить курс», «Добавить пользователя», «Удалить пользователя», «Редактировать информацию пользователя». Операция «Прочитать обращения из службы поддержки» доступна и без прав администратора, так как она может быть делегирована классу СПреподаватель.

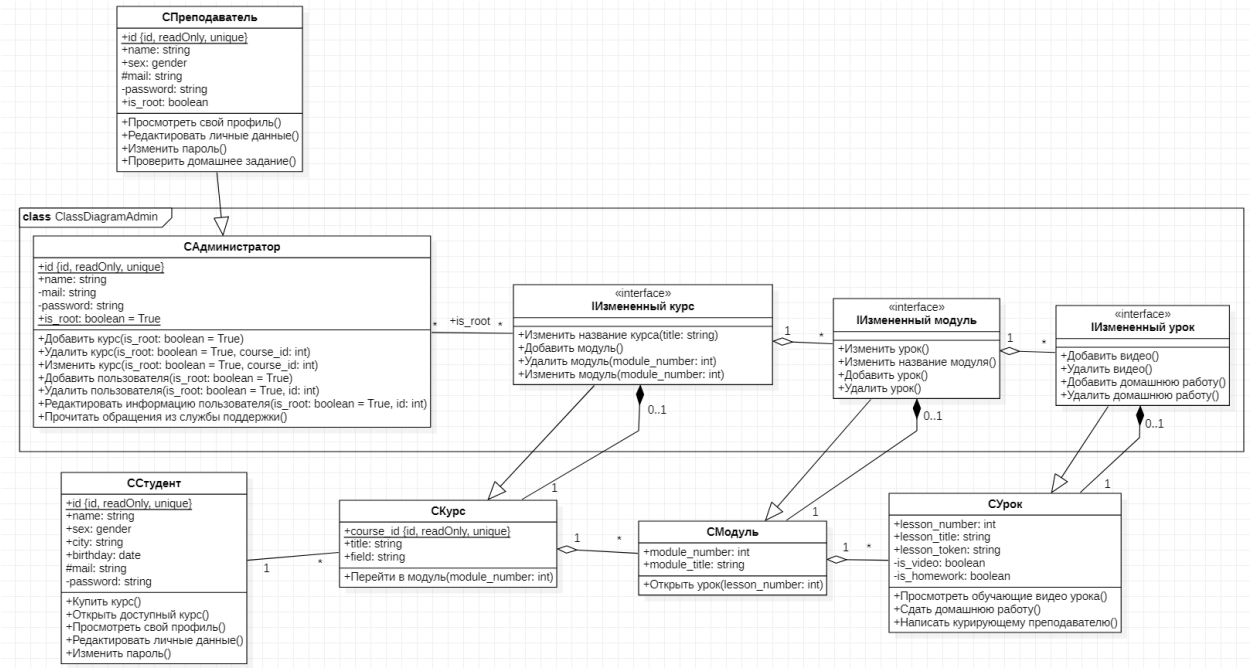

Рис. 21. Диаграмма классов для администратора

СПреподаватель – это дочерний класс одновременно класса САдминистратор и интерфейса IПосетитель, каждый экземпляр которого имеет следующие атрибуты: уникальный идентификатор id, имя name типа string, пол sex собственного типа gender, электронная почта mail типа string, пароль password типа string, атрибут is_root типа boolean. Так как данный класс является дочерним от САдминистратор, то каждый экземпляр класса также имеет следующий набор операций при условии is_root=True, что по сути своей означает наличие прав доступа администратора: «Добавить курс», «Удалить курс», «Изменить курс», «Добавить пользователя», «Удалить пользователя», «Редактировать информацию пользователя». Каждый экземпляр класса имеет следующий набор операций при любом значении параметра is_root: «Просмотреть свой профиль», «Редактировать личные данные», «Изменить

пароль», «Проверить домашнее задание». При вызове операции «Проверить домашнее задание» происходит переход к интерфейсу IДомашнее задание, которое создаётся по ходу операции «Сдать домашнюю работу» из класса СУрок.

Интерфейс IДомашнее задание имеет следующий набор операций: «Вернуть домашнее задание» возвращающее значение False, «Принять домашнее задание» возвращающее значение True, «Оставить комментарий».

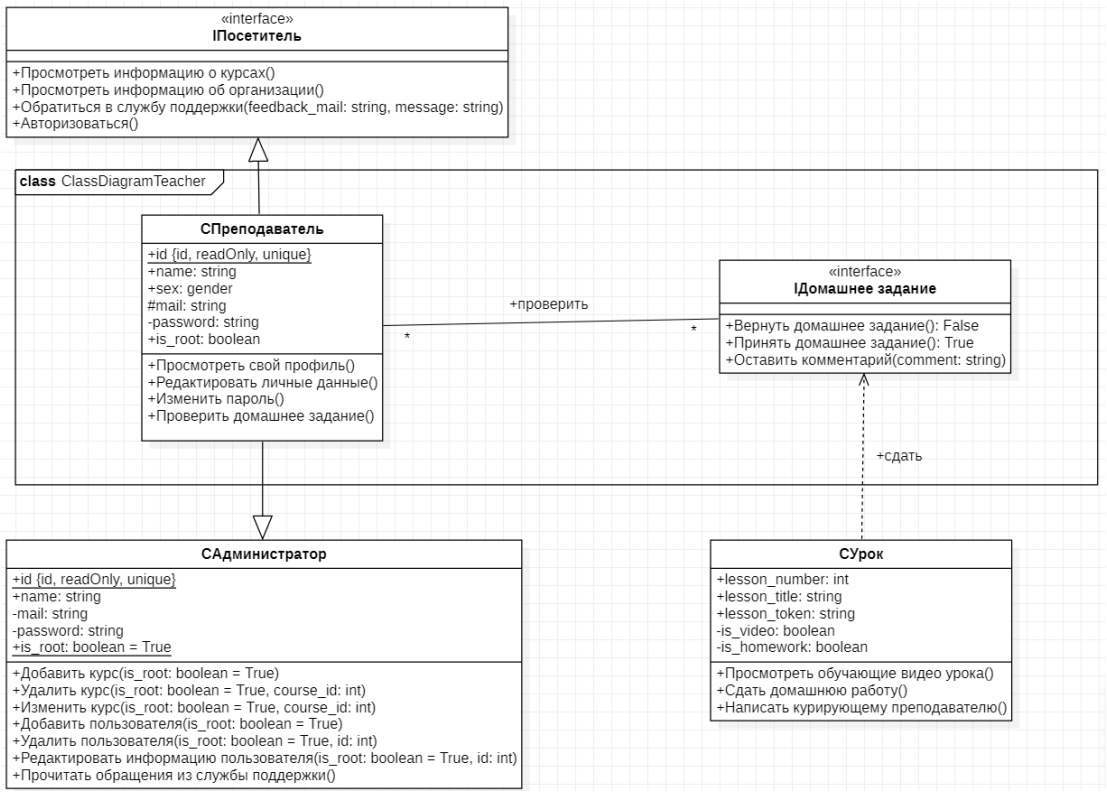

Рис. 22. Диаграмма классов для преподавателя

4.3 Диаграммы последовательности

На основе диаграмм использования были разработаны диаграммы последовательности действий для некоторых вариантов использования (рис. 23-25).

При авторизации посетителя сайта через интерфейс IПосетитель происходит переход к интерфейсу IАвторизация. Если у пользователя нет зарегистрированного аккаунта, то он проходит процесс создания аккаунта, при котором инициализируется экземпляр класса ССтудент. После этого пользователь может войти в систему посредством интерфейса IАвторизация.

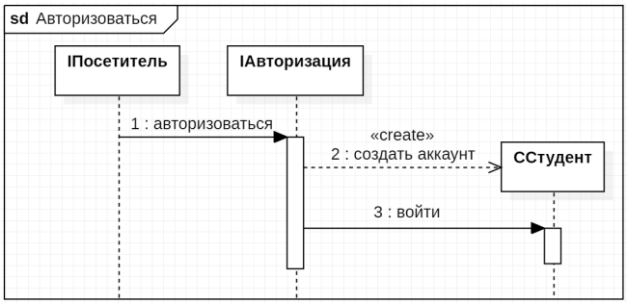

Рис. 23. Диаграмма последовательности действий для варианта использования «Авторизоваться»

Для проверки домашнего задания со стороны преподавателя класса СПреподаватель в первую очередь должна быть сдана домашняя работа определённого экземпляра класса СУрок, после чего создаётся интерфейс IДомашнее задание. После этого СПреподаватель может проверить IДомашнее задание и сделать одно из двух действий: вернуть задание на доработку, принять домашнее задание. Также вместе с любым решением, пользователь может оставить комментарий по своему усмотрению.

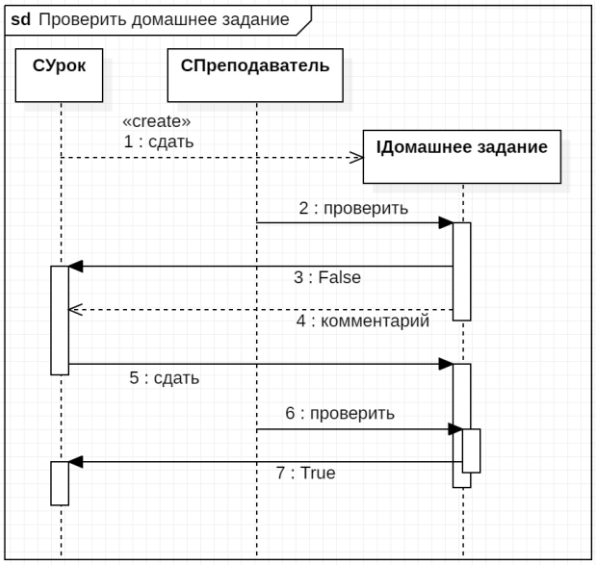

Рис. 24. Диаграмма последовательности действий для варианта использования «Проверить домашнее задание»

При решении изменить какой-либо курс САдминистратор совершает переход к интерфейсу IИзмененный курс, где следует выбрать что именно изменить. Если было принято решение об изменении названия курса, то в экземпляре класса СКурс задаётся новое значение атрибута title.

Рис. 25. Диаграмма последовательности действий для варианта использования «Изменить курс»

Заключение

В ходе проделанной работы была разработана модель информационной системы интернет-портала онлайн-образования на основе UML. Было составлено техническое задание. Также был проведён обзор строительные блоков UML и проанализированы общие механизмы их действия. Была выполнена практическая часть в виде построения одиннадцати диаграмм по модели информационной системы интернет-портала онлайн-образования. В них входят четыре диаграммы использования, четыре диаграммы классов и три диаграммы последовательности действий.

Список используемой литературы

1. Якобсон А., Буч Г., Рамбо Дж. Унифицированный процесс разработки программного обеспечения. – СПб.: Питер, 2002. – 496 с.

2. Мартин Фаулер. UML. Основы. Краткое руководство по стандартному языку объектного моделирования. 3-е издание. – М.: Символ-Плюс, 2013. – 192 с.

3. Якобсон А., Буч Г., Рамбо Дж. Краткая история UML // Язык UML. Руководство пользователя = The Unified Modeling Language Usere Guide. 2-е. – М.: ДМК Пресс, 2006. – 496 с

4. Rational and UML Partners. UML Specification version 1.1 (OMG document ad/97-08-11) – URL: https://www.omg.org/cgi-bin/doc?ad/97-08-11

5. Кватрани Т. Rational Rose 2000 и UML. Визуальное моделирование: Пер. с англ. – М.: ДМК Пресс, 2013 – 176 с.

6. Леоненков А. В. Нотация и семантика языка UML / Леоненков А. В. 2-е изд., исправ. – М.: Национальный Открытый Университет «ИНТУИТ», 2016. – 205 с. – URL: http://biblioclub.ru/index.php?page=book&id=429143

7. Леоненков А. В. Самоучитель UML 2. – СПб.: БХВ-Петербург, 2007. – 576 с.

8. Джозеф Шмуллер. Освой самостоятельно UML за 24 часа. 3-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2005. – 416 с.

9. Рамбо Дж., Блаха М. UML 2.0. Объектно-ориентированное моделирование и разработка. 2-е издание – СПб.: Питер, 2007 – 544 с.

10. Крэг Ларман. Применение UML 2.0 и шаблонов проектирования. Практическое руководство. 3-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2013. – 736 с.