Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

**ГПО АСУ-1101 «ОБЛАЧНАЯ ИНФОРМАЦИОННАЯ**

**СИСТЕМА ОБУЧЕНИЯ СТУДЕНТОВ»**

Отчёт по производственной практике

Студент гр. 434-1

\_\_\_\_\_\_\_\_\_\_\_ Ю.А. Богомолов

*Подпись*

Руководитель:

Преподаватель каф. АСУ

Доктор технических наук

\_\_\_\_\_\_\_\_\_\_\_ Профессор

*Оценка* М.Ю. Катаев

\_\_\_\_\_\_\_\_\_\_\_ *«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2017 г.*

*Подпись*

Оглавление

[Введение 3](#_Toc493772078)

[1 Раздел первый 4](#_Toc493772079)

[2 Раздел второй 5](#_Toc493772080)

[3 Раздел третий 6](#_Toc493772081)

[3.1 Подраздел первый 6](#_Toc493772082)

[3.2 Подраздел второй 7](#_Toc493772083)

[Заключение 8](#_Toc493772084)

# Введение

Производственная практика проходила на кафедре. В качестве темы задания была выдана тема ГПО АСУ-1101 – «Облачная информационная система обучения студентов». Данная работа является продолжением того, что было проделано в семестре в рамках ГПО и курсового проекта по Базам Данных. В связи с этим на начало работы уже имелся концепт разрабатываемой системы, а также модель базы данных. Поэтому на время практики была поставлена цель начать практическую реализацию. Исходя из этого были поставлены следующие задачи: определиться с инструментами, изучить их и начать разработку. Выполнение этих задач подготовит почву к дальнейшей разработке системы уже в учебном семестре в рамках проекта ГПО.

# 1 Краткое описание системы

Разрабатываемая система предназначена для того, чтобы упростить работу преподавателей, заинтересовать учащихся и в целом улучшить процесс обучения. Она должна уметь предоставлять преподавателям интерфейс для создания учебных материалов и выдачи их студентам, а студентам – интерфейс для создания решений к заданиям. Кроме того, она должна уметь автоматизировать процесс проверки решений учащихся с помощью специального программного обеспечения.

# 2 Процесс разработки

## 2.1 Подготовка к разработке

Для начала работы необходимо было определиться с языками и инструментами, с помощью которых бы велась разработка. Для начала были выбраны фреймворк Django и, соответственно, язык программирования Python 3 (далее просто Python). В Django по умолчанию предусмотрены три варианта СУБД: SQLite, MySQL и PostgreSQL. Из этих трёх вариантов самым оптимальным показался последний, из-за чего и был выбран.

Следующим этапом стало изучение Django и Python. Это заняло некоторое время, так как мои познания Python были достаточно посредственные, а к Django я прежде вообще не прикасался. Кроме того, пару дней пришлось также изучать CSS, чтобы потом верстать front end сайта. Впрочем, обучение совмещалось с разработкой системы, что не сильно снизило темп работы.

## 2.2 Реализация базы данных

Django позволяет упростить процесс реализации структуры базы данных следующим образом: предлагается описывать каждую сущность в качестве класса Python. Затем он преобразует созданные классы в команды выбранной СУБД. Пример такой сущности представлен в листинге 1.

Листинг 1. Код класса сущности ПОЛЬЗОВАТЕЛЬ

|  |
| --- |
| class User(models.Model):  """Сущность ПОЛЬЗОВАТЕЛЬ базы данных."""  name = models.CharField(max\_length=200)  job = models.CharField(max\_length=200)  description = models.TextField()  path = models.CharField(max\_length=256) |

У данного способа есть недостаток: нельзя создавать композитные первичные ключи. Если требуется сделать такое, то вместо первичного ключа необходимо задавать соответствующие атрибуты как «Уникальные в связке» (“Unique Together”). Пример такого решения представлен листингом 2.

Листинг 2. Решение проблемы отсутствия композитного ключа.

|  |
| --- |
| class Teacher(models.Model):  """  Сущность ПРЕПОДАВАТЕЛЬ базы данных.  ПОЛЬЗОВАТЕЛЬ, который может управлять проектами и выдавать задания.  """  user = models.ForeignKey(User, on\_delete=models.CASCADE)  project = models.ForeignKey(Project, on\_delete=models.CASCADE)  class Meta:  unique\_together = ('user', 'project') |

Кроме того, у сущности, где не указан явно первичный ключ, всегда будет уникальное автоинкрементируемое поле *id*, которое будет являться таковым. Впрочем, это скорее даже является плюсом в данном проекте, т.к. в подготовленной схеме базы данных большинство сущностей в качестве первичного ключа имели именно такое поле. Это можно увидеть на рисунке 1, демонстрирующем схему FA базы данных.

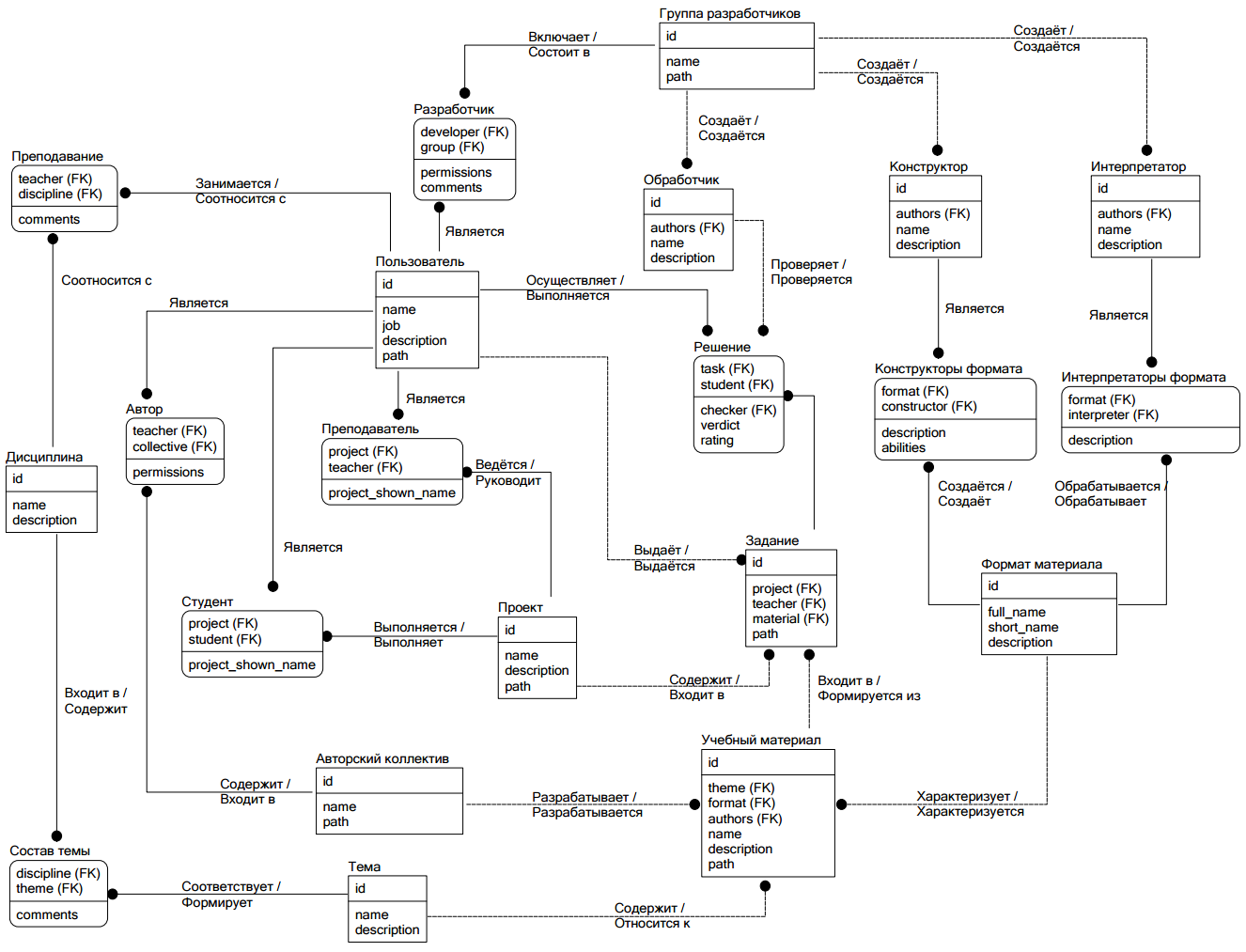


Рисунок 1. FA-модель базы данных

## 2.3 Создание представлений страниц

Django предлагает следующий способ обработки запросов и генерации страниц. Создаётся класс, который содержит методы обработки различных типов запросов (GET, POST и другие). Кроме того, он может извлекать из базы данных необходимые для этого данные. Так как сущностей в БД много, и для каждой было необходимо сделать страницу, а код получался однотипным, было решено написать генератор таких представлений, генератор для HTML-страниц, а также вспомогательные классы, оптимизирующие выборку данных. Код представления (View) страницы пользователя представлен листингом 3, а часть шаблона соответствующей HTML-страницы – листингом 4. Код, представленный в этих листингах большей частью был именно сгенерирован.

Листинг 3. Код представления страницы пользователя

|  |
| --- |
| class UserView(MyView):  """Представление страницы пользователя."""  template = 'core/user/user.html'  get\_context = cgg.make\_context\_getter(context={  'details': cgg.Get(User, pk='id'),  'teacher': {  'projects': cgg.Count(Teacher, user='id'),  'tasks': cgg.Count(Task, user='id'),  },  'student': {  'projects': cgg.Count(Student, user='id'),  'solutions': cgg.Count(Solution, user='id'),  },  'author': {  'collectives': cgg.Count(Author, user='id'),  },  'developer': {  'groups': cgg.Count(Developer, user='id'),  }  }, const\_getters={  'id': lambda \*\*kwargs: kwargs.get('pk'),  }) |

Листинг 4. Часть HTML-кода страницы пользователя

|  |
| --- |
| <table class = "info\_panel">  <tr><td>Name:</td><td>{{ details.name }}</td></tr>  <tr><td>Job:</td><td>{{ details.job }}</td></tr>  <tr><td>Description:</td><td>{{ details.description }}</td></tr>  </table>  <div class = "long-preview">  <a href = "{% url 'user/teacher' details.pk %}">  <span>Teacher</span>  {% if teacher %}  <ul>  {% if teacher.projects %}  <li>projects: {{ teacher.projects }}</li>  {% endif %}  {% if teacher.tasks %}  <li>tasks given: {{ teacher.tasks }}</li>  {% endif %}  </ul>  {% else %}  <p>You still have not started being teacher. Wanna try? Just click here!</p>  {% endif %}  </a>  </div> |

Кроме того, для оформления страниц потребовалось написать CSS-код, часть которого представлена листингом 5.

Листинг 5. Часть CSS-кода, используемого для отображения страниц системы

|  |
| --- |
| .info\_panel {  padding-left: 1em;  margin-bottom: 1em;  }  .preview, .long-preview {  display: inline-block;  background-color: #CCC;  width: 200px;  height: 80px;  margin-bottom: 1em;  margin-right: 1em;  padding-left: 0.5em;  padding-top: 0.5em;  padding-right: 1em;  font-family: Arial;  } |

# 3 Результаты работы

# Заключение

Заключение.