**Содержание**

|  |  |
| --- | --- |
|  | Стр. |
| Введение………………………………………………………………. | 3 |
| 1 Основные принципы разработки Web приложений на Django…..  2 Описание бизнес процесса по оформлению накладных в АО «ФНПЦ «ННИИРТ» и предложение по его оптимизации ………. | 4  9 |
| 3 Пример реализации Web приложения для оптимизации бизнес процесса по оформлению накладных на Django……………………. | 14 |
| Заключение……………………………………………………………. | 29 |
| Список литературы…………………………………………………… | 30 |

**Введение**

Деятельность компаний состоит из огромного количества повторяющихся бизнес-процессов, каждый из которых представляет собой последовательность действий и решений, направленных на достижение определенной цели. Прием заказа клиента, доставка товара клиенту, начисление зарплаты сотрудникам, оформление накладных и других документов. Эффективность деятельности компании в значительной степени определяется эффективностью реализации бизнес-процессов. В настоящее время даже в передовых, с точки зрения управленческих технологий, компаниях существуют возможности повышения эффективности отдельных подразделений и компании в целом путём оптимизации бизнес-процессов.

На крупном предприятии, имеющем большое количество контрагентов, в том числе по кооперации, таком как АО «ФНПЦ «ННИИРТ» каждый день оформляется большое количество накладных для передачи различных товарно-материальных ценностей.

В данной работе будет рассмотрен бизнес процесс по оформлению накладных в АО «ФНПЦ «ННИИРТ» и возможность оптимизации данного бизнес процесса с помощью web сервиса, разработанного на фреймворке Django.

1. **Основные принципы разработки Web приложений на Django.**

Django органично росла во время разработки реальных приложений, созданных командой разработки в Лоуренсе, штат Канзас, США. Она родилась в конце 2003 когда программисты газеты «Lawrence Journal-World», Эдриан Холовати (Adrian Holovaty) и Симон Виллисон (Simon Willison), начали использовать язык Python для разработки своих приложений. Команда World Online, ответственная за разработку и поддержку нескольких локальных новостных сайтов, процветала в среде разработчиков, диктуемой срочностью работы журналиста. Для сайтов, включая LJWorld.com, Lawrence.com и KUsports.com, журналисты и менеджеры требовали, чтобы новые возможности и целые приложения были разработаны максимально быстро, часто счёт шёл на дни или на часы. Таким образом, Симон и Эдриан создали среду разработки экономящую время, исходя из своих потребностей — это был единственный способ, с помощью которого они могли создавать управляемые приложения в жёстких рамках сроков.

Летом 2005, развив эту среду до такого состояния, что она поддерживала большинство сайтов World Online, команда разработчиков, включая Якоба Каплан-Мосса (Jacob Kaplan-Moss), решила выпустить среду в виде программного обеспечения с открытым исходным кодом. Они выпустили его в июле 2005 под именем Django в честь джазового гитариста Джанго Рэйнхарда (Django Reinhardt).

Сейчас Django является грамотным проектом с открытым исходным кодом с десятками тысяч пользователей, над которым работают разработчики по всей планете. Два автора из команды World Online («Великодушные пожизненные диктаторы», Эдриан и Якоб) всё ещё осуществляют общее руководство над развитием среды разработки, но теперь оно в большей степени зависит от совместных усилий команды разработчиков. World Online обеспечивает другие важные аспекты, такие как время разработчиков, маркетинговые материалы и хостинг/канал для сайта среды (http://www.djangoproject.com/).

Эта история уместна, потому что помогает объяснить два ключевых аспекта. Первый аспект — основное назначение Django. Так как Django родился в новостной среде, он предоставляет некоторые средства (такие как административный интерфейс), которые хорошо подойдут для контент-ориентированных сайтов, таких как Amazon.com, craigslist.org и washingtonpost.com, которые предоставляют динамическую информацию из базы данных. Существует разница между быть особенно эффективным в чём-то и быть неэффективным в другом.

Вторым аспектом является то, как происхождение Django сформировало культуру его сообщества открытого исходного кода. Так как Django был получен из реального кода, а не был академической разработкой или коммерческим продуктом, он полностью сфокусирован на решение проблем разработки, с которыми сталкивались и продолжают сталкиваться его авторы. В результате, Django постоянно совершенствуется. Основатели среды имеют свой интерес в том, чтобы Django экономил их время, создавал приложения лёгкие в обслуживании и хорошо работал под нагрузкой. При отсутствии других причин, разработчики мотивируются своими собственными эгоистичными желаниями сэкономить своё время и наслаждаться своей работой.

Django — свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Проект поддерживается организацией Django Software Foundation. MVC – схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.



Рисунок 1. Структура MVC шаблона.

Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других (например, Ruby on Rails). Один из основных принципов фреймворка — DRY (англ. Don't repeat yourself). Этот принцип разработки программного обеспечения, нацеленный на снижение повторения информации различного рода, особенно в системах со множеством слоёв абстрагирования. Принцип DRY формулируется как: «Каждая часть знания должна иметь единственное, непротиворечивое и авторитетное представление в рамках системы». Он был сформулирован Энди Хантом и Дэйвом Томасом в их книге The Pragmatic Programmer. Они применяли этот принцип к «схемам баз данных, планам тестирования, сборкам программного обеспечения, даже к документации». Когда принцип DRY применяется успешно, изменение единственного элемента системы не требует внесения изменений в другие, логически не связанные элементы. Те элементы, которые логически связаны, изменяются предсказуемо и единообразно.

В отличие от других фреймворков, обработчики URL в Django конфигурируются явно при помощи регулярных выражений. Для работы с базой данных Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных.

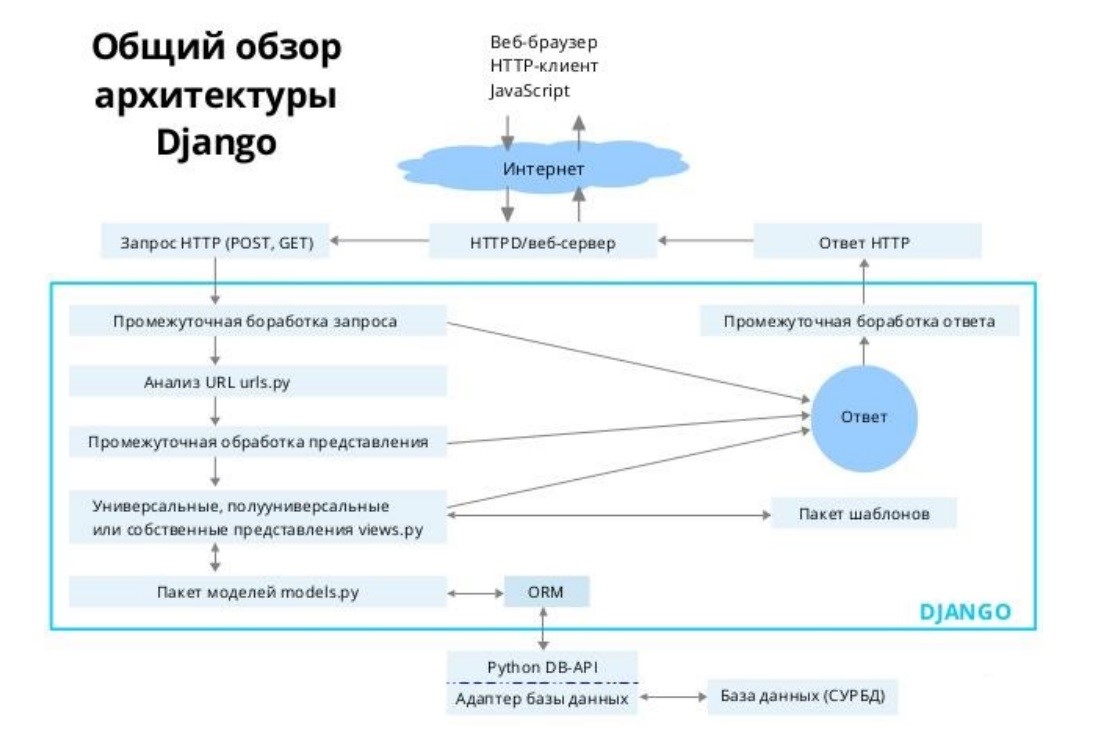


Рисунок 2. Общий обзор архитектуры Django

Рассмотрим более детально процесс создания Django-проектов.

Первоначальная разработка Django, как средства для работы новостных ресурсов, достаточно сильно отразилась на его архитектуре: он предоставляет ряд средств, которые помогают в быстрой разработке веб-сайтов информационного характера. Так, например, разработчику не требуется создавать контроллеры и страницы для административной части сайта, в Django есть встроенное приложение для управления содержимым, которое можно включить в любой сайт, сделанный на Django, и которое может управлять сразу несколькими сайтами на одном сервере. Административное приложение позволяет создавать, изменять и удалять любые объекты наполнения сайта, протоколируя все совершённые действия, и предоставляет интерфейс для управления пользователями и группами.

Django проект может включать в себя несколько приложений, каждое из которых состоит из четырех основных компонентов.

1. Модель данных (models): данные являются сердцевиной любого современного Web-приложения. Модель - важнейшая часть приложения, которое постоянно обращается к данным при любом запросе из любой сессии. Любая модель является стандартным Python классом. Объектно-ориентированный маппер (ORM) обеспечивает таким классам доступ непосредственно к базам данных. Если бы не было ORM, программисту пришлось бы писать запросы непосредственно на SQL. Модель обеспечивает облегченный механизм доступа к слою данных, инкапсулирует бизнес-логику. Модель не зависит от конкретного приложения. Данными можно манипулировать даже из командной строки, не используя при этом Web-сервер.

2. Представление (views): выполняют разнообразные функции, в том числе контролируют запросы пользователя, выдают контекст в зависимости от его роли. View - это обычная функция, которая вызывается в ответ на запрос какого-то адреса (URL) и возвращает контекст.

3. Шаблоны: являются формой представления данных. Шаблоны имеют свой собственный простой метаязык и являются одним из основных средств вывода на экран.

4. URL: механизм внешнего доступа к представлениям (views). Встроенные в URL регулярные выражения делают механизм достаточно гибким. При этом одно представление может быть сконфигурировано к нескольким урлам, предоставляя доступ различным приложениям. Здесь поддерживается философия закладок: URL становятся самодостаточными и начинают жить независимо от представления.

В дистрибутив Django также включены приложения для системы комментариев, синдикации RSS и Atom, «статических страниц» (которыми можно управлять без необходимости писать контроллеры и представления), перенаправления URL и другое.

1. **Описание бизнес процесса по оформлению накладных в АО «ФНПЦ «ННИИРТ» и предложение по его оптимизации.**

Несмотря на достаточно развитую инфраструктуру АО «ФНПЦ «ННИИРТ» и наличия корпоративной информационной сети бизнес процесс по оформлению накладных в АО «ФНПЦ «ННИИРТ» не оптимизировался уже более 10 лет. Ответственным за оформление накладных является экономист отдела сбыта. В отдел сбыта поступают служебные записки на оформление накладных от всех подразделений организации. Процесс оформления накладных достаточно прост, но занимает почти все рабочее время из-за большого количества заявок, за год оформляется порядка 1500 накладных.

В основном в организации используются две формы накладных: М-15 и Торг-12.

Накладная на отпуск материалов на сторону. Форма М-15. Форма утверждена постановлением Госкомстата РФ от 30.10.1997 N 71а.

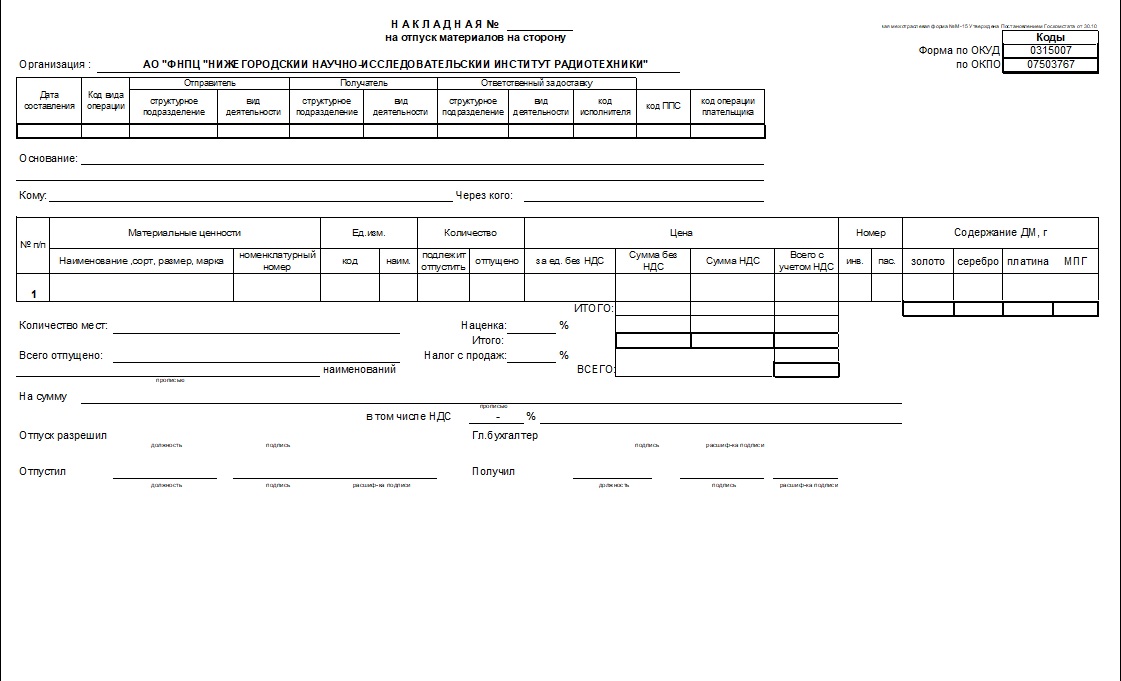


Рисунок 3. Образец накладной формы М-15.

При передаче материальных ценностей из одного предприятия в другое, либо представителям компании, которые находятся вне территории фирмы, используется накладная на отпуск материалов на сторону. Для этих целей применяется типовая форма М-15. Выписывается данная накладная в двух экземплярах для организации, отпускающей товар, и для получателя соответственно. Основанием для оформления накладной служит служебная записка на имя начальника отдела сбыта. Важным фактом является обязательное наличие доверенности у получателя ТМЦ, которую он предъявляет работнику подразделения, передающего имущество, для получения товарных ценностей.

Для оформления накладной используется форма файла, заполняемая в Microsoft Excel. Требуется указать дату выписки и номер документа, код операции (при необходимости) и наименование подразделения, поставляющего товар. Затем вносятся сведения о материальных ценностях (наименование и характеристика), наименование и код единицы измерения, номенклатурный номер (при наличии). В 7-8 строках заполняется количество материалов, необходимых для отпуска по сопроводительному документу, и количество отпущенных ТМЦ. Указывается стоимость единицы без учета налога. В 10 строке подсчитывается общая стоимость отпущенных материалов. Затем прописывается общая сумма НДС по всем материалам и общая сумма ТМЦ с учетом налога.

Ниже заполняется инвентарный номер товара, номер технического паспорта (только для драгоценных металлов и камней) и номер записи из карты учета. В конце формы прописью указывается число, сумма и НДС всех отпущенных материалов. Подписывается накладная сотрудником, разрешившим отпуск ТМЦ, отпустившим материалы, главным бухгалтером предприятия и представителем получателя.

Для передачи материальных ценностей в другие организации оформляются накладные, служащие подтверждением факта передачи имущества.

Форма № ТОРГ-12 «Товарная накладная» утв. Постановлением Госкомстата от 25.12.1998 № 132.

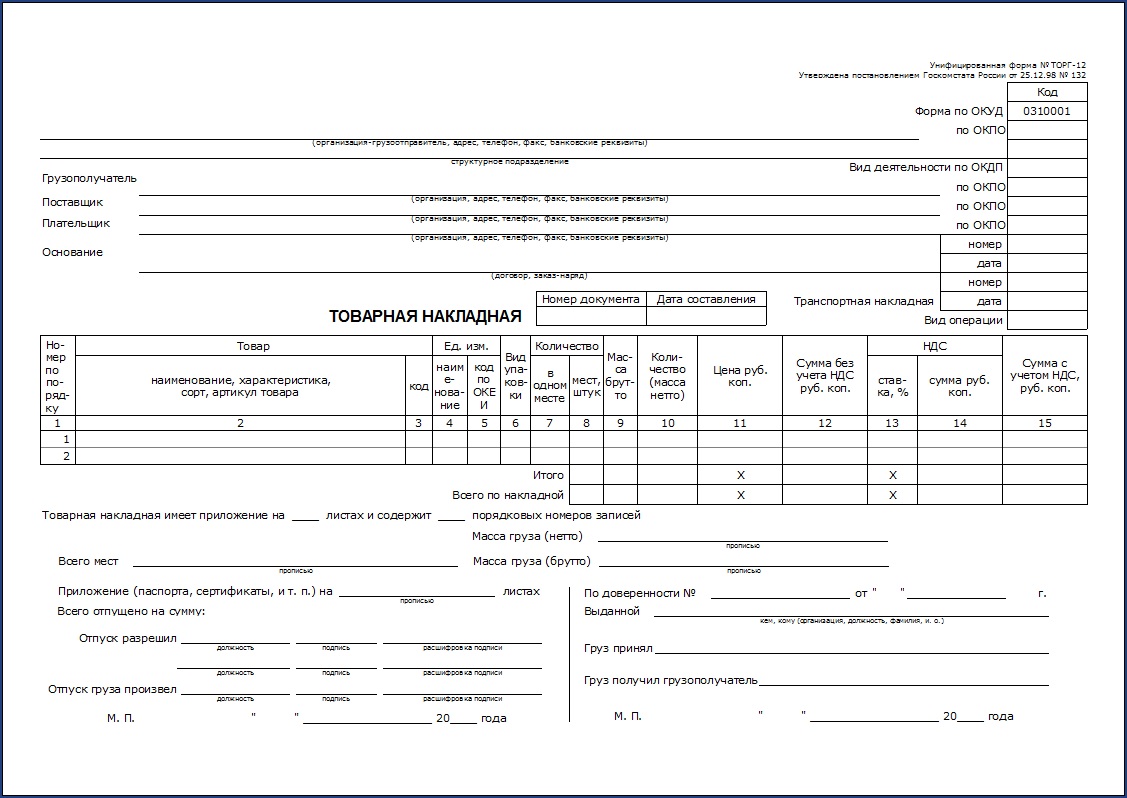


Рисунок 4. Образец формы Торг-12.

Накладная ТОРГ-12 применяется для оформления продажи (отпуска) товарно-материальных ценностей сторонней организации. Накладная составляется в 2 экземплярах. Первый экземпляр остается у организации, сдающей товарно-материальные ценности (т. е. у поставщика), и является основанием для их списания. Второй экземпляр передается сторонней организации (покупателю) и является основанием для оприходования этих ценностей.

Для того, чтобы накладная ТОРГ-12 могла считаться первичным учетным документом, необходимо контролировать наличие в ней, в частности, следующих обязательных реквизитов (ч. 2 ст. 9 Федерального закона от 06.12.2011 № 402-ФЗ):

* дата составления накладной;
* наименование составителя накладной (поставщика);
* наименование отпускаемых ТМЦ, их натуральное и денежное измерение;
* наименование должностей лиц, ответственных за отпуск ТМЦ и оформление накладной, а также фамилии, инициалы и подписи таких лиц.

Чтобы из накладной № ТОРГ-12 явно следовало, что это именно продажа ТМЦ, целесообразно заполнять реквизит «Основание». Здесь можно указать наименование договора, по которому отпускаются ТМЦ. Ведь ТОРГ-12 может быть оформлена и на возврат товаров – тогда в основании указывается, что это возврат.

Важно контролировать, чтобы со стороны получателя ТОРГ-12 была подписана уполномоченными лицами. К примеру, руководителем организации или лицом, действующим на основании доверенности. В последнем случае к накладной необходимо приложить такую доверенность.

Разберем подробно существующий в АО «ФНПЦ «ННИИРТ» бизнес процесс оформления накладных.

Подразделение, инициирующее передачу имущества направляет в отдел сбыта служебную записку, в которой в обязательном порядке указывает всю информацию, необходимую для заполнения формы накладной экономистом отдела сбыта:

1. Номер служебной записки
2. Структурное подразделение отправитель
3. Основание для отправки (письмо, рекламационный акт, протокол тех. Совещания и т.д.)
4. Сокращенное наименование предприятия получателя, его адрес
5. Перечень имущества, его количество, единицы измерения
6. Стоимость имущества (с НДС или без).
7. Количество мест
8. Дата вывоза.
9. Исполнитель и контактный телефон.

Далее экономист открывает форму накладной и заполняет её в соответствии со служебной запиской, после чего распечатывает и подписывает у руководства организации.

При данном построении бизнес процесса имеются следующие недостатки:

* Возможность появления ошибок. Экономисту отдела сбыта приходится еще раз вручную набирать тот же текст, который напечатан в служебной записке. При этом так же имеет место человеческий фактор – экономист может совершать опечатки при наборе данных.
* Потеря времени. Много времени уходит для перепечатывания текста из служебной записки в форму накладной и зависит от количества товаров (в некоторых случая может занимать более часа).
* Неполное указание данных. Служебная записка пишется в свободной форме, зачастую данные для оформления накладной указываются не полностью, при этом экономисту отдела сбыта приходится повторно обращаться в подразделение, отправляющее продукцию, за недостающими данными.

Для оптимизации бизнес процесса по оформлению накладных, достаточно создать веб сервис по подаче заявок в отдел сбыта на оформление накладных через единый информационный портал АО «ФНПЦ «ННИИРТ». Страница подачи заявок на оформление накладных будет включать в себя обязательные для заполнения поля и передавать их содержимое в базу данных. Далее экономист отдела сбыта через административную панель будет экспортировать данные из базы данных в формате .xls для последующего автоматического заполнения формы накладной. Данный способ сможет исключить ошибки при оформлении накладных экономистом, сэкономить время и избавить от необходимости приносить напечатанные служебные записки в отдел сбыта. Подробно реализацию данного проекта разберем следующей главе.

1. **Пример реализации Web приложения на Django, для оптимизации бизнес процесса по оформлению накладных.**

Для создания проекта понадобится визуальная среда разработки. Наиболее часто используемые для django средами являются: Atom и PyCharm. В данном примере будет использоваться PyCharm 2019.1.2 x64.

После запуска программы необходимо создать проект, указав тип проекта django и имя проекта. Затем PyCharm создаст структуру проекта автоматически.

mysite/

manage.py

mysite/

\_\_init\_\_.py

settings.py

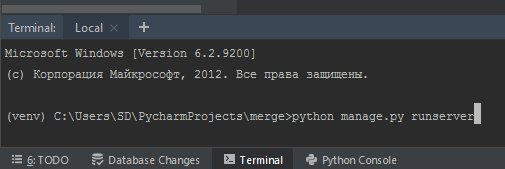
urls.py

wsgi.py

Рассмотрим эти файлы:

* Внешний каталог mysite/ – это просто контейнер для вашего проекта. Его название никак не используется Django, и вы можете переименовать его во что угодно.
* manage.py: Скрипт, который позволяет вам взаимодействовать с проектом Django. Подробности о manage.py читайте в разделе django-admin и manage.py.
* Внутренний каталог mysite/ - это пакет Python вашего проекта. Его название – это название пакета Python, которое вы будете использовать для импорта чего-либо из проекта (например, mysite.urls).
* mysite/\_\_init\_\_.py: Пустой файл, который указывает Python, что текущий каталог является пакетом Python. (Читайте о пакетах в официальной документации Python, если вы новичок в Python.)
* mysite/settings.py: Настройки/конфигурация проекта. Раздел Настройки Django расскажет вам все о настройках проекта.
* mysite/urls.py: Конфигурация URL-ов для вашего проекта Django. Это “содержание” всех Django-сайтов. Вы можете прочитать о конфигурации URL-ов в разделе Менеджер URL-ов.
* mysite/wsgi.py: Точка входа вашего проекта для WSGI-совместимых веб-серверов. Подробности читайте в разделе Развёртывание с WSGI.

Чтобы проверить, что все заработало выполним команду в терминале:



$ python manage.py runserver

Теперь, когда сервер запущен, перейдем на страницу <http://127.0.0.1:8000/> в браузере. Увидим страницу с “Welcome to Django”. Все работает.

Каждое приложение Django состоит из пакета Python, который следует некоторым соглашениям. Django содержит команду, которая создает структуру для нового приложения, что позволяет вам сосредоточиться на написании кода, а не на создании каталогов.

Создадим приложение, в котором будет работать веб сервис по подаче заявок. Я назвал его “face”. Для создания приложения face необходимо в терминале выполнить следующую команду:

python manage.py startapp face

Эта команда создаст каталог face:

face/

\_\_init\_\_.py

admin.py

apps.py

migrations/

\_\_init\_\_.py

models.py

tests.py

views.py

Эти файлы являются частью приложения.

В нашем приложении создания заявок, мы создадим две модели: orders и products, связанные между собой. Orders содержит подразделение, причину, кому отправлен товар и дату, products содержит наименование продукции, его количество, единицу измерения и цену. Каждая модель представлена классом, унаследованным от django.db.models.Model. Каждая модель содержит несколько атрибутов, каждый из которых отображает поле в таблице базы данных. То есть каждая модель – является таблицей в базе данных, а её содержимое – столбцами данной таблицы.

Записываем код в merge/models.py

from django.db import models  
  
  
class Order(models.Model):  
 date = models.DateField(verbose\_name='Дата вывоза')  
 subdivision = models.CharField(max\_length=50, verbose\_name='Подразделение')  
 reason = models.CharField(max\_length=50, verbose\_name='Причина')  
 where = models.CharField(max\_length=50, verbose\_name='Куда')  
  
 class Meta:  
 verbose\_name = 'Заказ'  
 verbose\_name\_plural = 'Заказы'  
  
  
class Products(models.Model):  
 order = models.ForeignKey(Order, on\_delete=models.CASCADE)  
 name = models.CharField(max\_length=50, verbose\_name='Наименование')  
 price = models.FloatField(verbose\_name='Цена')  
 quantity = models.FloatField(verbose\_name='Количество')  
  
 PIECES = 'шт'  
 KILOGRAMMS = 'кг'  
 METERS = 'м'  
 KOMPLEKT = 'кт'  
 MEASUREMENTS = (  
 (PIECES, 'шт'),  
 (KILOGRAMMS, 'кг'),  
 (METERS, 'м'),  
 (KOMPLEKT, 'кт'),  
 )  
 measurement = models.CharField(  
 max\_length=2,  
 choices=MEASUREMENTS,  
 default=PIECES, verbose\_name='ед. изм'  
 )  
  
 class Meta:  
 verbose\_name = 'Товар'  
 verbose\_name\_plural = 'Товары'  
  
 def is\_upperclass(self):  
 return self.measurement in (self.PIECES, self.KOMPLEKT)  
  
 def \_\_str\_\_(self):  
 return self.name

Разберем данный код. class Order и class Products – являются моделями, это таблицы в базе данных, а их поля – столбцы. Так же выбираются типы данных в столбцах: Char.Field – текстовое поле, оно имеет обязательный параметр max\_length, который обязателен для заполнения, иначе программа выдаст ошибку перед запуском сервера. DateField и FloatField – тип данных для даты и для чисел с плавающей запятой соответственно, verbose\_name и verbose\_name\_plural – служит для выведения наименования данного поля как в форме, так и в административной панели в единственном и множественном числе соответственно. Для создания выпадающего списка с единицами измерений используется словарь.

Активация моделей. Эта небольшая часть кода моделей предоставляет Django большое количество информации, которая позволяет Django:

* Создать структуру базы данных (CREATE TABLE) для приложения.
* Создать Python API для доступа к данным объектов Order и Products.

Но первым делом мы должны указать нашему проекту, что приложение face установлено.

Приложения Django “подключаемые”: вы можете использовать приложение в нескольких проектах и вы можете распространять приложение, так как они не связаны с конкретным проектом Django.

Для того чтобы наше приложение работало необходимо сообщить что приложение face установлено и добавить запись в файле settings.py в пункте INSTALLED\_APPS.

merge/settings.py

1

INSTALLED\_APPS **=** [

'face',

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

]

По умолчанию, INSTALLED\_APPS содержит следующие приложение, которые предоставляются Django:

django.contrib.admin – интерфейс администратора. Скоро мы его будем использовать.

django.contrib.auth – система авторизации.

django.contrib.contenttypes – фреймверк типов даных.

django.contrib.sessions – фреймверк сессии.

django.contrib.messages – фреймверк сообщений.

django.contrib.staticfiles – фреймверк для работы со статическими файлами.

Эти приложения включены по умолчанию для покрытия основных задач.

Теперь Django знает, что необходимо использовать приложение face. Давайте выполним следующую команду:

python manage.py makemigrations face

Выполняя makemigrations, вы говорите Django, что внесли некоторые изменения в ваши модели (в нашем случае мы создали несколько новых) и хотели бы сохранить их в миграции.

Миграции используются Django для сохранения изменений ваших моделей (и структуры базы данных) - это просто файлы на диске. Вы можете изучить миграцию для создания ваших моделей, она находится в файле face/migrations/0001\_initial.py. Но их формат удобен для чтения на случай, если вы захотите внести изменения.

В Django есть команда, которая выполняет миграции и автоматически обновляет базу данных - она называется migrate. Теперь, выполните команду [migrate](https://djbook.ru/rel1.9/ref/django-admin.html#django-admin-migrate), чтобы создать таблицы для этих моделей в базе данных:

python manage.py migrate

Команда migrate выполняет все миграции, которые ещё не выполнялись, (Django следит за всеми миграциями, используя таблицу в базе данных django\_migrations) и применяет изменения к базе данных, синхронизируя структуру базы данных со структурой ваших моделей.

Миграции - очень мощная штука. Они позволяют изменять ваши модели в процессе развития проекта без необходимости пересоздавать таблицы в базе данных. Их задача изменять базу данных без потери данных. Три инструкции по изменению моделей:

* Внесите изменения в модели (в models.py).
* Выполните python manage.py makemigrations чтобы создать миграцию для ваших изменений
* Выполните python manage.py migrate чтобы применить изменения к базе данных.

Две команды необходимы для того, чтобы хранить миграции в системе контроля версий. Они не только помогают вам, но и могут использоваться другими программистами вашего проекта.

Создание суперпользователя. Первым делом необходимо создать пользователя, который может заходить на интерфейс администратора. Выполните следующую команду:

python manage.py createsuperuser

Введите имя пользователя и нажмите Enter.

sername: admin

Теперь введите email:

Email address: admin@example.com

И наконец введите пароль.

assword: \*\*\*\*\*\*\*\*\*\*

Password (again): \*\*\*\*\*\*\*\*\*

Superuser created successfully.

Запускаем сервер для разработки. Интерфейс администратора включен по умолчанию. Давайте запустим встроенный сервер для разработки и посмотрим на него. Если сервер не запущен, выполните:

python manage.py runserver

Откроем “/admin/” локального домена в браузере – например, http://127.0.0.1:8000/admin/. Вы должны увидеть страницу авторизации интерфейса администратора:

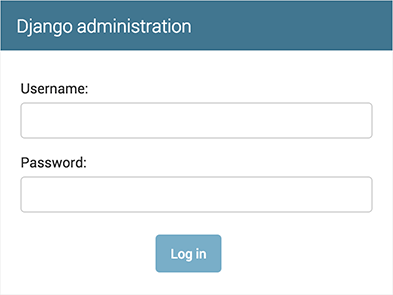


Рисунок 5. Авторизация в панели администратора django.

Заходим в интерфейс администратора. Вы должны увидеть несколько разделов: группы и пользователи. Они предоставлены приложением авторизации Django django.contrib.auth. Добавим приложение в интерфейс администратора, так как оно не отображается.

merge/admin.py

from django.contrib import admin  
from .models import \*

admin.site.register(Order)

admin.site.register(Products)

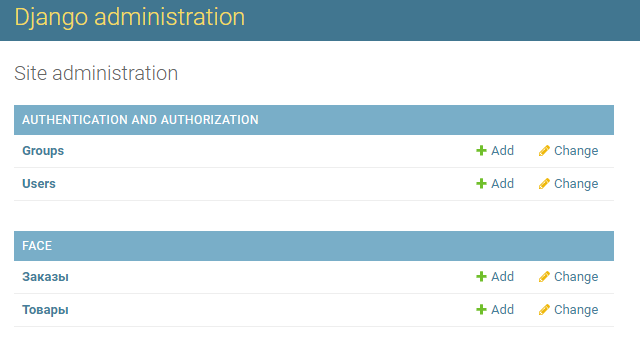


Рисунок 6. Созданные модели в панели администратора.

Далее создадим файл forms.py в приложении face. В этом файле будут прописаны формы, которые будут вставлены в форму html документа.

from django.forms import ModelForm

from face.models import Order, Products

class OrderForm(ModelForm):

class Meta:

model = Order

fields = ['date', 'subdivision', 'reason', 'where']

exclude = ['']

class ProductsForm(ModelForm):

class Meta:

model = Products

fields = ['name', 'price', 'quantity', 'measurement']

exclude = ['']

Разберем данный код. Дня начала мы импортируем модули форм Django и созданные модели из файла models. В классах форм прописываем модель к которой относится форма, так же поля, которые будут выведены для заполнения в html форме. Обязательным параметром является запись exclude или include. Эти записи говорят о том какие поля необходимо включить или исключить, в нашем случае ничего не исключается, поэтому значение остается пустым.

Перейдем к самому объемному из файлов – admin.py, отвечающего за панель администратора. Ранее в нем уже зарегистрировали модели для их отображения в панели администратора, теперь добавим новые классы для более удобного отображения информации в панели администратора, а так же функции для экспорта данных в формат excel.

class ProductInOrderInline(admin.TabularInline):  
 model = Products  
 extra = 0  
  
  
class OrderAdmin (admin.ModelAdmin):  
 actions = [export\_xls\_o]  
 list\_display = [field.name for field in Order.\_meta.fields]  
 inlines = [ProductInOrderInline]  
  
 class Meta:  
 model = Order  
  
  
admin.site.register(Order, OrderAdmin)  
  
  
class ProductsAdmin (admin.ModelAdmin):  
 actions = [export\_xls\_p]  
 list\_display = [field.name for field in Products.\_meta.fields]  
  
 class Meta:  
 model = Products  
  
  
admin.site.register(Products, ProductsAdmin)

Для того чтобы вложить таблицу products в таблицу order создаем класс ProductInOrderInline с шаблоном отображения admin.TabularInline. Теперь мы сможем видеть, какие товары были вложены в конкретную заявку (Order).

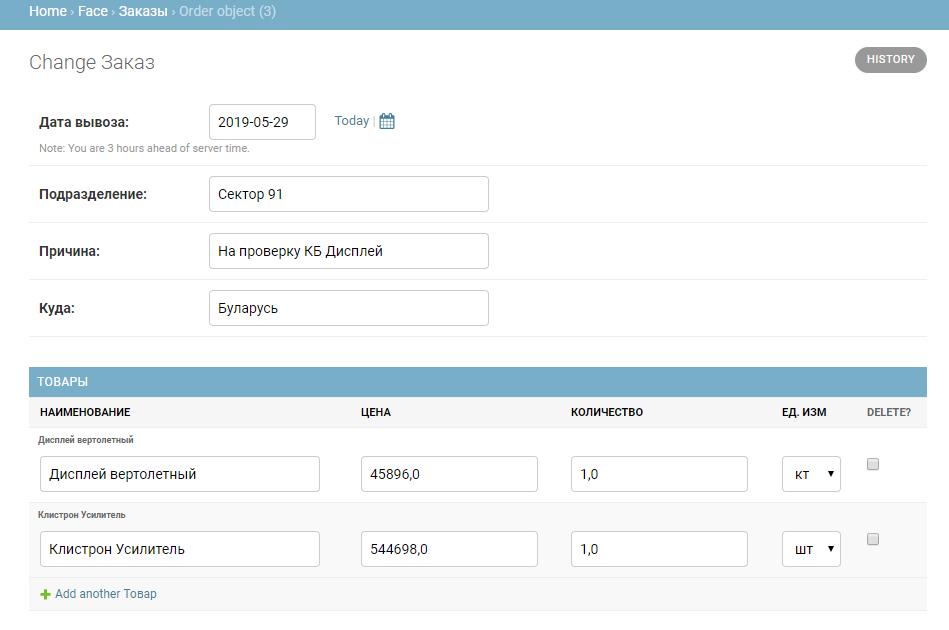


Рисунок 6. Отображение вложенной таблицы Товары в таблицу Заказов.

Для написания функции экспорта данных в .xls файл понадобится установить дополнительную библиотеку “xlwt” позволяющую создавать таблицы Microsoft Excel. Для установки данной библиотеки воспользуемся стандартным установщиком pip. Открываем терминал и пишем pip install xlwt, если у вас возникают проблемы с ssl сертификатами необходимо изменить запись на «pip install --trusted-host pypi.org --trusted-host files.pythonhosted.org <package\_name>» заменив package\_name на xlwt. Данный вид записи пригодится для установки любых стандартных библиотек. После установки можно импортировать xlwt для дальнейшего использования. Так как мы используем две модели, то необходимо будет прописать две функции, одну для модели order другую для products. Разберем код для экспорта таблицы products.

def export\_xls\_p(modeladmin, request, queryset):

import xlwt

response = HttpResponse(content\_type='application/ms-excel')

response['Content-Disposition'] = 'attachment; filename=product\_model.xls'

wb = xlwt.Workbook(encoding='utf-8')

ws = wb.add\_sheet("Products")

row\_num = 0

columns = [

(u"ID", 2000),

(u"name", 6000),

(u"quantity", 8000),

(u"price", 8000),

(u"measurement", 8000),

]

font\_style = xlwt.XFStyle()

font\_style.font.bold = True

for col\_num in range(len(columns)):

ws.write(row\_num, col\_num, columns[col\_num][0], font\_style)

# set column width

ws.col(col\_num).width = columns[col\_num][1]

font\_style = xlwt.XFStyle()

font\_style.alignment.wrap = 1

for obj in queryset:

row\_num += 1

row = [

obj.pk,

obj.name,

obj.quantity,

obj.price,

obj.measurement,

]

for col\_num in range(len(row)):

ws.write(row\_num, col\_num, row[col\_num], font\_style)

wb.save(response)

return response

export\_xls\_p.short\_description = u"Export XLS"

Указываем название функции, далее его будет нужно добавить действие класса ProductAdmin: actions = [export\_xls\_p], в OrderAdmin: actions = [export\_xls\_o] соответственно, имя файла задает responce['Content-Disposition'], в данном случае каждый новый файл будет иметь название product\_model.xls. В columns перечисляем все столбцы таблицы, которые хотим экспортировать в файл. Благодаря данной функции в панели администратора появляется новое действие, Export\_XLS.

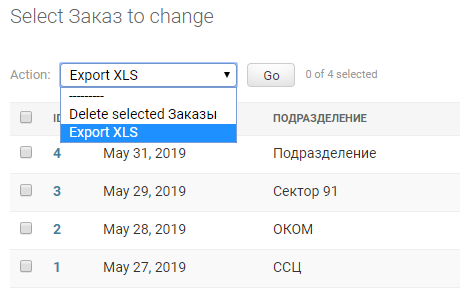


Рисунок 7. Действие по экспорту .xls файла из панели администратора.

Таким образом после выбор необходимых позиций, их можно экспортировать из панели администратора в excel файл для дальнейшего копирования в поля накладной, независимо от формы Торг-12 или М-15, так как обе формы заполняются в Excel.

За вывод представлений в браузер отвечает файл views.py, в котором прописывается функция, вызываемая в ответ на запрос какого-то адреса (URL) и возвращающая контекст. Адреса хранятся в файле приложения face/urls.py. В данном приложении код файла urls.py следующий:

from django.urls import path

from face import views

urlpatterns = [

path('', views.new\_order, name='home'),

В пути ничего не указано, поэтому функция представления new\_order вызывается при переходе на начальный адрес сайта. При необходимости адрес можно изменить. К примеру, если путь изменить на path('sendingorder/', views.new\_order, name='home'), то при переходе на начальный адрес сайта будет выводиться ошибка (404) page not found, представление будет отображаться в случае перехода на http://127.0.0.1:8000/'sendingorder/. Разберем код файла представления views.py:

from face.forms import OrderForm, ProductsForm

from django.shortcuts import render

def new\_order(request):

if request.method == "POST":

order\_form = OrderForm(request.POST)

products\_form = ProductsForm(request.POST)

if order\_form.is\_valid() and products\_form.is\_valid():

order = order\_form.save()

products = products\_form.save(False)

products.order = order

products.save()

return render(request, 'index.html', locals())

После импорта форм объявляется имя функции и два вложенных условия, первый проверяет метод запроса, второй валидность форм, после чего сохраняет записи в базу данных. Для отображения в браузере используется файл index.html. Html файлы хранятся в папке проекта templates. Для вывода форм в структуре html кода прописываем название наших форм в следующем виде:

<form method="post" action="">

{% csrf\_token %}

{{ order\_form.as\_p }}

{{ products\_form.as\_p }}

<div>

<button type="submit" class="btn btn-primary">Отправить заявку</button>

</div>

</form>

Таким образом страница для подачи заявки будет выглядеть следующим образом.

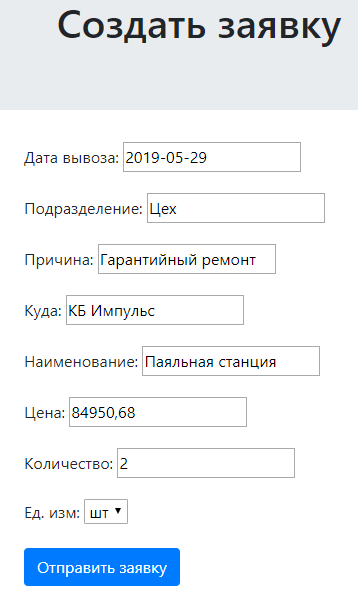


Рисунок 8. Форма заявки на странице портала.

**Заключение**

В начале работы был разобран принцип построения web приложений с помощью Django. Основными преимуществами данного фреймворка являются его модульность – возможность подключения и отключения различных приложений к проекту, возможность управлять сразу несколькими сайтами на одном сервере и встроенная административная панель.

В данной работе был разобран текущий алгоритм бизнес процесса по оформлению накладных в АО «ФНПЦ «ННИИРТ», выявлены недостатки данного алгоритма и предложен проект по его оптимизации. Для оптимизации данного бизнес процесса был разработан проект web приложения на свободном фреймворке для Web приложений Django, который можно разместить на внутреннем портале (сайте) АО «ФНПЦ «ННИИРТ».

Были рассмотрены преимущества при работе через web приложение, а именно: сокращение рабочего времени, затрачиваемого на данный бизнес процесс вследствие исключения необходимости повторного набора текста, а так же снижение количества ошибок благодаря минимизации человеческого фактора. Была разработана единая форма подачи заявок, с обязательным заполнением всех полей для недопущения пропусков в информации, необходимой для оформления накладной. Для дальнейшего перехода на электронное взаимодействие между подразделениями, по созданию заявок на оформление накладных, необходимо создать дополнительную страницу на портале, разработать всю необходимую документацию и внести изменения в стандарт организации.

**Список литературы**

1. Официальный сайт Django [Электронный ресурс]. – https://www.djangoproject.com/ (дата обращения 01.04.2019)
2. DjangoTricks [Электронный ресурс]. – https://djangotricks.blogspot.com/2013/12/how-to-export-data-as-excel.html – (дата обращения 22.05.2019)
3. Документация django на русском [Электронный ресурс]. –https://djbook.ru/ – (дата обращения 06.04.2019)
4. Википедия – свободная энциклопедия [Электронный ресурс]. – http://wikipedia.org . – (дата обращения 05.04.2019)
5. Адриан Головатый, Джейкоб Каплан-Мосс, Django подробное руководство / Адриан Головатый, Джейкоб Каплан-Мосс – Символ-Плюс, 2018 – 549 c
6. Aidas Bendoraitis, Django 2 Web Development Cookbook - Third Edition / Aidas Bendoraitis, Jake Kronika – Packt, 2018 – 544 c
7. Python 3 для начинающих [Электронный ресурс]. – https://pythonworld.ru/ – (дата обращения 09.04.2019)
8. Карпова, Т.С. Базы данных. Модели, разработка, реализация [Электронный ресурс] : учебное пособие / Т. С. Карпова. - Электрон. дан. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 403 c
9. Уэсли Чан, Джефф Форсье, Пол Биссекс, Django. Разработка веб-приложений на Python / Уэсли Чан Мосс – Символ-Плюс, 2013 – 442 c
10. Официальный сайт Bootstrap [Электронный ресурс]. – https://getbootstrap.com/ – (дата обращения 20.04.2019).