Здесь располагается разрастающаяся инструкция по тому, как создать свой сайт на основе Django, gUnicorn, nginx и MySQL на Ubuntu 14.04 LTS Desktop x64, запущенной под VirtualBox. Так что первым делом, прежде чем приступить к описанию установки и настройки несущих компонентов сайта, нужно установить и настроить Ubuntu под виртуалкой. Выбрал 14.04, так как она LTS, и свежая, а значит, с ориентиром на будущее, хотя тут я вообще профан, и ваще это дело вкуса).

**Часть 1. Установка и настройка всего необходимого.**

Создание виртуальной машины с Ubuntu:

1. Скачиваем и устанавливаем VirtualBox для своей операционной системы, тут всё предельно просто, никаких махинаций не требуется, единственное, что удобно под образы дисков (с которых можно устанавливать гостевые ос на машины) выделить отдельную папку (у меня это <путь к папке VirtualBox VMs>/Distrs).
2. Создаём новую виртуальную машину, выбираем Linux, 64-bit, Ubuntu, тип виртуального жёсткого диска – на усмотрение. Вот [тут](http://mirspo.narod.ru/vbox/ch05s02.html) описаны основные принципы выбора, а вообще, на эту тему можно легко найти кучу инфы, но это в нашем случае не принципиально.
3. После того, как машина будет создана, настраиваем её (кнопка Настроить): я выделил под виртуалку 2048Мб ОЗУ, 2 ядра CPU, 128Мб видеопамяти, сетевой адаптер – не принципиально, будет это NAT или сетевой мост, так как нам нужен только доступ в Интернет во внешнюю среду. Сетевой мост может быт полезным, если захочется с другого компьютера домашней сети проверить что-нибудь на виртуальной машине.  
   Важно: приоритет загрузки: сначала CD-ROM, потом жёсткий диск, ибо мы установим убунту с образа диска. Чтобы сделать это, в настройках Носители контроллером IDE ставим привод CD/DVD, сбоку кнопочка для выбора образа диска, выбираем скачанный образ нашей операционной системы в формате iso, и ставим галочку около «Живой диск».
4. Запускаем виртуальную машину. Далее идёт обычный процесс установки Ubuntu. Устанавливаем, выбирая Erase disk and install (или как-то так). После завершения установки и перезагрузки, может выскочить сообщение о необходимости вытащить диск с убунтой, жмакаем Enter, машина перезагрузится, и всё будет норм.  
   После установки не забудьте убрать галочку в Система –> Загрузка –> CD-ROM.

Конфигурирование виртуальной машины:

1. Первым делом установим все обновления:  
   sudo apt-get update  
   sudo apt-get upgrade
2. Затем установим расширения для гостевой ОС, чтобы увеличить разрешение экрана:

sudo apt-get install virtualbox-guest-utils  
Затем перезагружаем машину, затем на окне с гостевой ОС, Жмём Устройства –> Подключить образ диска дополнений гостевой ОС, далее выполняем всё, что будет выскакивать.  
Затем выключаем виртуальную машину (заодно ставим жёсткий диск первым в очерёдности загрузки в настройках виртуальной машины), а затем запускаем её снова, и разрешение экрана само поменяется, иначе просто в настройках дисплея меняем на желаемое, я, например, выставил 1600х1200, чтобы побольше влезало на мелкий экран виртуалки, правда и больше ресурсов будет жраться.  
И, да, кстати: после установки дополнений гостевой ОС есть смысл включить двунаправленный буфер обмена в настройках виртуальной машины, тогда можно будет копипастить туда-сюда, это удобно

1. Чтобы компьютер мог проигрывать большинство форматов аудио и видео, установим Ubuntu Restricted Extras:

sudo apt-get install ubuntu-restricted-extras

1. По своему вкусу устанавливаем всё остальное, что может пригодиться в будущем, например, для разработчиков на Java может пригодиться Oracle JDK, скачать его можно с официального сайта Oracle. А вообще, вот [тут](http://www.linuxrussia.com/2014/04/ubuntu-1404.html) неплохо описано, что можно сделать со свежеустановленной Ubuntu 14.04 LTS. Также некоторым может понадобиться Git.
2. Некоторые договорённости об установке ПО.
3. /opt – директория для стороннего ПО, не требующего установки (распаковал – юзай), например, PyCharm
4. Для целых наборов пакетов использовать виртуальное окружение, общие средства разработки ставить общесистемно

Теперь приступим к установке всего, что нужно для разработки на Django.

Сразу отмечу, что [тут](http://m05quit0.ru/linux/ustanovka-dopolnitel-ny-h-paketov-ubuntu-dlya-razrabotki-na-django/) есть некоторая полезная инфа о том, что устанавливать на Убунту для Джанго (но сначала лучше пройтись по моей инструкции).

1. Сначала установим на всю систему python, если он не установлен, у меня уже была установлена версия 2.7.6
2. Затем установим и настроим virtualenv:  
   sudo apt-get install python-virtualenv

Виртуальное окружение нужно, чтобы иметь на одной девелоперской системе несколько изолированных средств, так, мы создадим среду для разработки на Django, куда установим, собственно, Django, а так же gUnicorn и pip. Возможно, установим и питон, если его там ещё не будет. Но, по идее, python, pip и easy\_install уже будут изолированно доступны. По virtualenv [тут](http://adw0rd.com/2012/6/19/python-virtualenv/#.VQ1kFI6sU00) хорошо написано (отсюда я и брал инструкции по установке и настройке).  
Настройка виртуального окружения:  
а. Создаём папку в системе, где будут располагаться окружения (здесь – в домашней папке создаём папку с окружениями):

mkdir ~/Enviroments

б. Создаём новое окружение djenv в папке Enviroments:  
cd Enviroments  
virtualenv djenv

в. Читаем ссылку, где хорошо написано про virtualenv, в частности, про работу: что надо активировать среду перед работой в ней, и деактивировать после

1. Установим и настроимMySQL:  
    sudo apt-get install mysql-server python-mysqldb libmysqlclient-dev mysql-client

Кодировка при работе с Django – UTF-8, но БД может не везде выставлять её, так что явно в конфиге укажем это:  
а. Откроем в редакторе конфигу:   
sudo gedit /etc/mysql/my.cnf

б. В секцию [mysqld] добавим следующие строки:  
skip-character-set-client-handshake  
character-set-server=utf8  
init-connect='SET NAMES utf8'  
collation-server=utf8\_general\_ci

в. В секциях [client] и [mysqldump] добавим строку:  
default-character-set=utf8

Готово, теперь перезапустим сервер MySQL:  
sudo service mysql restart

Далее нужно будет создать базу данных, создать пользователя, и присвоить ему все привилегии для использования данной БД.  
 а. Запускаем mysql:  
 sudo mysql –p  
 Вводим пароль для root, который указали при установке mysql. Далее проверим, применились ли настройки, которые мы меняли в конфиге. Для этого введём команду:  
SHOW VARIABLES LIKE 'char%';  
Должно выдаться такое:  
character\_set\_client utf8   
character\_set\_connection utf8   
character\_set\_database utf8   
character\_set\_filesystem binary   
character\_set\_results utf8   
character\_set\_server utf8   
character\_set\_system utf8   
character\_sets\_dir /usr/share/mysql/charsets/

При вводе команды:  
SHOW VARIABLES LIKE 'coll%';  
Должно выдаться такое:  
collation\_connection utf8\_unicode\_ci  
collation\_database utf8\_unicode\_ci  
collation\_server utf8\_unicode\_ci

Обратите внимание, \_ci означает, что при сравнении строк не учитывается регистр. Возможно вам следует использовать utf8\_general\_cs или utf8\_bin, если это будет важно в будущем (по крайней мере, теперь мы знаем, куда заглянуть, если надо будет это поменять). Полезную инфу, если что, брал [отсюда](http://djbook.ru/examples/48/) и [отсюда](http://help.ubuntu.ru/wiki/mysql).

б. Создаём базу данных serbd:  
CREATE DATABASE vzaledb CHARACTER SET utf8;

в. Создаём пользователя ser ('ser'@'localhost') с паролем password:  
CREATE USER 'ser'@'localhost' IDENTIFIED BY 'password';

г. Предоставляем созданному пользователю доступ к созданной базе данных:  
GRANT ALL PRIVILEGES ON vzaledb.\* TO 'ser'@'localhost'; (или просто ser;)  
Имя до точки – имя базы данных, звёздочка после точки означает, что доступ будет ко всем таблицам базы данных vzaledb.

д. После завершения махинаций с правами доступа необходимо выполнить данную команду:  
FLUSH PRIVILEGES;

Готово, БД на данном этапе пока что можно считать настроенной. Инфу брал [тут](https://www.digitalocean.com/community/tutorials/mysql-ru), тут же можно почерпнуть ещё полезностей.

1. Теперь установим nginx:  
   sudo apt-get install nginx
2. Теперь установим Gunicorn:  
   а. Активируем виртуальное окружение, в которое поставим gunicorn:  
   source ~/Enviroments/djenv/bin/activate  
   Слева в каждой строчку командной строки теперь будет «(djenv)».  
   б. Установим gunicorn:  
   pip install gunicorn

в. Деактивируем виртуальное окружение:  
deactivate

Пока что с nginx и gunicorn закончим, и перейдём к установке и настройке Django.

1. Установка Django.

а. Активируем виртуальное окружение:  
source ~/Enviroments/djenv/bin/activate

б. Установим Django:  
pip install Django

Заметим, что после установки, переходя к созданию проекта, виртуальное окружение не деактивируем, ведь Джанго находится в нём, а мы с ним будем работать.

Настройка Django.

а. В удобной для хранения проекта папке (у меня это ~/Sites/onDjango) создаём новый проект с именем vzale:  
django-admin.py startproject vzale

Получилась следующая структура проекта:

vzale /

manage.py

vzale /

\_\_init\_\_.py

settings.py

urls.py

wsgi.py

б. Настроим БД. Для этого откроем, например, в gedit, settings.py:  
cd vzale  
gedit vzale /settings.py

Находим DATABASES, и изменяем его согласно [инструкциям](https://docs.djangoproject.com/en/1.7/intro/tutorial01/) в туториале (в которых, кстати, много полезного, и по которым я и делаю данные пункты инструкции) и [тут](https://docs.djangoproject.com/en/1.7/ref/settings/#std:setting-DATABASES):  
DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'vzaledb',

'USER': 'ser',

'PASSWORD': 'password',

'HOST': '127.0.0.1',

}

}

Заметим, что имя юзера – без суффикса @localhost. Данные поля обязательны, есть также опциональный поля, о них в документации написано подробнее.

в. Теперь немного магии: нужно поставить некоторые девелопмент тулзы для питона, причём поставить их в виртуальном окружении:  
sudo apt-get install python-dev  
Затем поставить MySQL-python:  
pip install MySQL-python

Теперь можно выполнить миграцию:  
python manage.py migrate

г. Готово! Теперь запустим сервер:  
python manage.py runserver  
Зайдя по адресу <http://127.0.0.1:8000> в браузере, увидим приветствие от сервера.

д. Все статические файлы соберём в одном месте, в папке ~/Sites/onDjango/vzale /static/, для этого в settings.py:  
STATIC\_ROOT = '/home/ser/Sites/onDjango/vzale /static/'STATIC\_URL = '/static/'  
Про STATIC\_URL – пока что неизвестно, но, по идее, неважно, ибо статику будет раздавать nginx. Затем выполняем операцию сбора статики:  
python manage.py collectstatic

е. Создадим суперпользователя для входа в админку Django:  
python manage.py createsuperuser  
ser  
[proser93@mail.ru](mailto:proser93@mail.ru)  
vzale32admin47

1. Теперь надо разобраться с IDE для Python. Выбрал PyCharm, ставить будем глобально, на систему: а то вдруг пригодится. Так что не забыть про deactivate, чтобы выйти из виртуального окружения.  
   а. Для PyCharm требуется JDK. [Установим](http://help.ubuntu.ru/wiki/java) последнюю 8 версию:  
   sudo add-apt-repository ppa:webupd8team/java

sudo apt-get update  
echo oracle-java8-installer shared/accepted-oracle-license-v1-1 select true | sudo /usr/bin/debconf-set-selections

sudo apt-get install oracle-java8-installer  
б. Скачиваем PyCharm в папку /opt и распаковываем:  
cd /opt  
sudo wget <http://download.jetbrains.com/python/pycharm-community-4.0.5.tar.gz>  
sudo tar -xzvf pycharm-community-4.0.5.tar.gz  
в. Архив за ненадобностью можно удалить:  
sudo rm pycharm-community-4.0.5.tar.gz

г. Установка завершена. Добавим для удобства ярлык на рабочий стол:  
gedit ~/.local/share/applications/pycharm.desktop  
В файле пишем следующее и сохраняем:  
[Desktop Entry]

Name=PyCharm

Comment=Среда разработки PyCharm

GenericName=PyCharm

Keywords=PyCharm;Python;Django;IDE;

Exec= /opt/pycharm-community-4.0.5/bin/pycharm.sh

Terminal=false

Type=Application

Icon= /opt/pycharm-community-4.0.5/bin/pycharm.png

Categories=

Path= /opt/pycharm-community-4.0.5/bin  
Всё. Теперь в поиске в графической оболочке будет PyCharm. Затем открываем и настраиваем начальные вещи.  
Сюда при необходимости будут вписаны инструкции по интеграции с виртуальным окружением, если это понадобится. А так, при открытии отдельных файлов с кодом, просто в Settings – Project Interpretier указать python в папке bin, которая находится в корне виртуального окружения.

1. Настройка Gunicorn

Снова активируем виртуальное окружение, если деактивировали его. Затем из папки, где находится manage.py, проверим, что через Gunicorn всё работает:  
gunicorn vzale.wsgi  
В браузере должно всё быть, как после успешной проверки Django. Это значит, что работает такая схема: Пользователь <-> Gunicorn <-> Django.

1. Теперь настроим nginx. Для начала проверим, что он работает: в браузере надо проверить, работает ли localhost:80. Там должна быть страничка приветствия от nginx.  
   Теперь настроим конфигурационный файл. По умолчанию, конфигурационный файл называется nginx.conf и расположен в каталоге /usr/local/nginx/conf, /etc/nginx или /usr/local/etc/nginx. В папке sites-available создаём новый файл vzale, открываем его для редактирования, и вписываем:  
   server {

server\_name localhost;

location /images/ {

alias /home/ser/Sites/onDjango/ vzale /data/images/;

}

location /files/ {

alias /home/ser/Sites/onDjango/ vzale /data/media/files/;

}

location = / {

proxy\_pass http://127.0.0.1:8000;

}

location / {

alias /home/ser/Sites/onDjango/ vzale /data/www/index.html;

}

}

Файл default удаляем, а в папке sites-enabled создаём символическую ссылку на файл vzale из sites-available.  
Перезапускаем nginx:  
sudo service nginx restart  
Затем запускаем gunicorn, и проверяем ,что всё работает)

Вариант конфига:

server {

server\_name localhost;

root /home/ser/Sites/onDjango/ vzale /data;

location / {

try\_files $uri @proxy\_to\_app;

}

location @proxy\_to\_app {

proxy\_pass http://127.0.0.1:8000;

}}

**Часть 2. Разработка своего веб-проекта (сайта).**

Здесь будут инструкции по установке дополнительного вспомогательного ПО, его настройке, и использованию. Когда Django проект уже существует, junicorn запущен, а nginx проксирует запросы и умеет выдавать статику, пришло время создать (сверстать) страницы сайта.

1. Выбор тематики сайта. Я выбрал создание портала для пользователей GRANIT – научно-исследовательской сети. Там будут располагаться новости по развитию данной инфраструктуры, а также будут репозитории с ПО, и т.д. Также там будут веб-средства для пользователей, страница регистрации, страница личного профиля, вики, и мб ещё трак. ***Полный состав допишу здесь позднее.***
2. Выбор структуры сайта. Пока что сайт называется vzale. Полное доменное имя я ещё не придумал и не выбрал. Это пока что и не нужно. Итак, список минимум 5 страниц, которые нужно сверстать для начала:  
   1. Главная страница – 3 колонки, верхний и нижний разделы, левая колонка – ссылки на структуру сайта (профиль и другие разделы ,и т.д.), средняя – контент текущего раздела, правая – опции для текущего раздела (например, если открыт профиль, то справа будут ссылки на генерацию новой или загрузку своей пары ssh-ключей). ***Полный состав допишу здесь позднее.***  
   2. Страница регистрации – формы для ввода информации, предполагается изначально ручная регистрация (т.к. доступ в сеть будет ограничен научными кругами). ***Поля здесь допишу позднее.***3. Страница профиля – пока что внешний вид не придумал, будет примерно как на веб-портале portal.geni.net. Возможно, будет добавлена аватарка, ник, также будут личные данные, список файлов (ключи и сертификат), список срезов, ссылки на использование юзер-тулзов. ***Допишу здесь полный состав позднее.***4. Страница поста – как главная страница, только в средней колонке будет содержание одного поста, снизу комментарии. ***Допишу здесь полный состав позднее.***

5. Страница с картой ресурсов, мониторинг и т.д. – как resource view во Flukes. ***Допишу здесь полный состав позднее.***

1. Средства для вёрстки.  
   1. Bootstrap.   
   ПОКА ЧТО НЕ ИСПОЛЬЗОВАТЬ ([Здесь](http://bootstrap-3.ru/getting-started.php) описаны основные моменты установки. Необходимо прежде установить все зависимости. В идеале нужно юзать Bower, но это в будущем, вот инструкции (примерные, [тут](https://www.youtube.com/watch?v=Zx8f390IyOg&index=2&list=PLY4rE9dstrJwP_JUTts9AtMVtJlJVGxWK) есть видеоуроки по Bower)  
   а. Bower. Требует установки Node, npm и Git:  
   sudo apt-get install npm node git  
   Установка Bower:  
   sudo npm install -g bower)  
   Скачиваем [отсюда](https://github.com/twbs/bootstrap/releases/download/v3.3.2/bootstrap-3.3.2-dist.zip) готовую версию, либо на сайте указываем свою конфигурацию, компилируем и скачиваем. Распаковываем скачанный архив в директорию проекта, я распаковал bootstrap в папку static/vendors, так как там будут css стили, и шрифты и прочее, и нахождение в одной директории (static) упростит встраивание в html код.  
   Потребуется jQuery, можно встраивать гугловскую, но лучше локальную, скачиваем [отсюда](http://jquery.com/download/), и распаковываем в удобную папку. Я распаковал в папку jQuery в папке vendors в папке static
2. Первичная вёрстка index.html. На [этом](https://www.youtube.com/watch?v=9akF4udEueA&list=PLY4rE9dstrJwP_JUTts9AtMVtJlJVGxWK&index=1) видео показан первый урок работы с bootstrap, где показано, как встраивать css в html. Крайне советую просмотреть данные уроки по bootstrap. Не забываем в случае локальной jQuery использовать её при встраивании в html
3. Кратко об уроках на youtube: первый – вводный, как испытать, что работает, вставить готовые шаблоны. Второй урок о таблицах, как при масштабировании выстраивать элементы (например, изображения), как настраивать взаимное расположение элементов на разных размерах экрана, и как скрывать или отображать те или иные элементы на различных размерах экранов.

Создание своего приложения

1. Создаём приложение с именем blog:

python manage.py startapp blog

1. Создаём модели

В models.py вписываем все необходимые модели  
Для работы с моделями нам понадобится задать для них методы, это лучше сделать добавлением своего менеджера для каждой модели:  
http://djbook.ru/rel1.8/ref/models/instances.html

1. Сохраняем миграции моделей:  
   python manage.py makemigrations
2. Выполняем миграцию в БД:  
   python manage.py migrate
3. Загрузить модели в админку, для этого добавляем их в admin.py в папке blog
4. Создать скрипт рандомного заполнения моделей для проверки  
   Faker – библиотека для генерации рандомных данных:  
   pip install fake-factory