**Дипломен проект**

**Държавен изпит за придобиване на трета степен на професионална квалификация – част по теория на професията**

**по професия код 481030 „Приложен програмист”**

**специалност код 4810301 „Приложно програмиране”**

**Тема:** Инсталиране на училищен (учебен) web-сървър за Python/Django.

Изготвил:

Салих Кесаджи

Ръководител:

инж. Георги Бориков

Съдържание

[2 Увод 3](#_Toc133448715)

[3 Основна част 4](#_Toc133448716)

[3.1 Какво е web сървър 4](#_Toc133448717)

[3.2 Как работят уеб сървърите? 4](#_Toc133448718)

[3.3 URL адрес 5](#_Toc133448719)

[3.4 HTTP 6](#_Toc133448720)

[3.5 Клиент (потребителският агент) 8](#_Toc133448721)

[3.6 Функциите на HTTP 9](#_Toc133448722)

[3.7 HTTP и връзки 9](#_Toc133448723)

[3.8 HTML 10](#_Toc133448724)

[3.9 IP адрес 11](#_Toc133448725)

[3.10 Какво е DHCP и DHCP сървър? 13](#_Toc133448726)

[3.11 Каква е основната функция на уеб сървърите? 15](#_Toc133448727)

[3.12 DNS 15](#_Toc133448728)

[3.13 Какво се случва при зареждането на даден уебсайт? 17](#_Toc133448729)

[3.14 Уеб сървър като услуга 17](#_Toc133448730)

[3.15 Коя операционна система да изберем за web сървъра ни. 18](#_Toc133448731)

[3.16 Основните разлики между Windows и Linux. 19](#_Toc133448732)

[3.17 Защо Linux Ubuntu? 21](#_Toc133448733)

[3.18 Apache или IIS? 22](#_Toc133448734)

[3.19 Виртуални среди и пакети 23](#_Toc133448735)

[3.20 Инсталиране на сървъра 24](#_Toc133448736)

[3.21 Инсталиране на Python и Apache 27](#_Toc133448737)

[3.22 Инсталиране на MySQL база данни 28](#_Toc133448738)

[3.23 Инструкция за Python/Django 29](#_Toc133448739)

[4 Заключение 37](#_Toc133448740)

[5 Източници и използвана литература 40](#_Toc133448741)

# Увод

Уеб сървърът е софтуерно приложение, което предоставя платформа за хостване и обслужване на уебсайтове и уеб базирани приложения по интернет. Когато потребител въведе URL адреса на уебсайт в своя уеб браузър, заявката се изпраща до уеб сървъра, който отговаря, като изпраща съдържанието на уебсайта обратно към браузъра на потребителя. Уеб сървърите обикновено използват HTTP (Hypertext Transfer Protocol) или HTTPS (HTTP Secure), за да комуникират с уеб браузъри.

Уеб сървърите могат да работят на различни операционни системи, включително Windows, Linux и macOS. Те могат също да бъдат хоствани на физически или виртуални сървъри или в облачно базирани среди като Amazon Web Services (AWS) или Microsoft Azure. Някои популярни уеб сървъри включват Apache HTTP сървър, Nginx и Microsoft IIS (Интернет информационни услуги).

Обучението по специалности приложно програмиране, системно програмиране и КТТ предполага разработване на сайтове и други приложения. За да могат учениците от изброените специалности в ПГЕЕ — Банско да работят в среда, максимално приближена до реалната, е необходимо е да се инсталира собствен училищен WEB сървър. За целта се решат следните задачи:

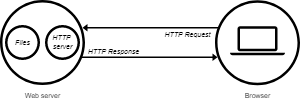
* Да се направи преглед на най-популярните WEB сървъри.
* Да се избере подходящо решение за операционна система и WEB сървър.
* На предоставен от училището сървърен хардуер (компютър) да се инсталират избраната ОС, софтуер, реализиращ функционалността на WEB сървър и актуална версия на Python.
* Да се състави инструкция за разполагане на сървъра на приложения, разработи на база платформата Python/Django

# Основна част

## Какво е web сървър

Терминът уеб сървър може да се отнася до хардуер, софтуер, или и двете. От страна на хардуера, уеб сървърът е компютър, който съхранява софтуера на уеб сървъра и компонентните файлове на уебсайта (например HTML документи, изображения, CSS таблици със стилове и JavaScript файлове). Уеб сървърът се свързва с интернет и поддържа физически обмен на данни с други устройства, свързани с мрежата. От страна на софтуера уеб сървърът включва няколко части, които контролират как уеб потребителите имат достъп до хоствани файлове. Като минимум това е HTTP сървър . HTTP сървърът е софтуер, който разбира URL адреси (уеб адреси) и HTTP (протоколът, който вашият браузър използва за преглед на уеб страници). HTTP сървърът може да бъде достъпен чрез имената на домейни на уебсайтовете, които съхранява. Той доставя съдържанието на тези хоствани уебсайтове до устройството на крайния потребител. На най-основно ниво, когато браузърът се нуждае от файл, който се хоства на уеб сървър, браузърът изисква файла чрез HTTP. Когато заявката достигне правилния (хардуерен) уеб сървър, (софтуерният) HTTP сървър приема заявката, намира искания документ и го изпраща обратно към браузъра, също чрез HTTP. (Ако сървърът не намери искания документ, вместо това връща отговор: Грешка 404. )

## Как работят уеб сървърите?

Софтуерът на уеб сървъра е достъпен чрез имената на домейни на уебсайтовете и осигурява доставката на съдържанието на сайта до искащия потребител. Софтуерната страна също се състои от няколко компонента, с поне HTTP сървър. HTTP сървърът може да разбира HTTP и URL адреси. Когато уеб браузър , като Google Chrоme или Microsoft Edge, се нуждае от файл, който се хоства на уеб сървър, браузърът ще поиска файла чрез HTTP заявка. Когато заявката бъде получена от уеб сървъра, HTTP сървърът ще приеме заявката, ще намери съдържанието и ще го изпрати обратно към браузъра чрез HTTP.По-конкретно, когато браузър поиска страница от уеб сървър, процесът ще следва поредица от стъпки. Първо, потребителя ще посочи URL в адресната лента на уеб браузъра. След това уеб браузърът ще получи IP адреса на името на домейна - или чрез превод на URL чрез DNS (която описвам подробно на стр.15) или чрез търсене в неговия кеш . Това ще доведе браузъра до уеб сървър.

След това браузърът ще поиска конкретния файл от уеб сървъра чрез HTTP заявка. Уеб сървърът ще отговори, изпращайки на браузъра исканата страница отново чрез HTTP. Ако заявената страница не съществува или ако нещо се обърка, уеб сървърът ще отговори със съобщение за грешка. След това браузърът ще може да покаже уеб страницата.

Уеб сървърът е софтуерна програма, която работи на компютър и отговаря за хостването на уеб съдържание и обслужването му на клиенти, които го поискат. Когато потребител въведе URL в уеб браузър, браузърът изпраща заявка до уеб сървъра за съдържанието, свързано с този URL. След като толкова пъти споменахме URL адрес и HTTP нека да разберем какво точно означават те.

## URL адрес

Уеднаквен локатор на ресурси (URL) е адресът на ресурси в интернет. URL адресът показва местоположението на ресурса, както и протокола, използван за достъп до него.URL адресът съдържа следната информация:

* Протоколът, използван за достъп до ресурса
* Местоположението на сървъра (независимо дали чрез IP адрес или име на домейн)
* Номер на порта на сървъра (незадължително)
* Местоположението на ресурса в структурата на директория на сървъра
* Идентификатор на фрагмент (незадължителен)

Известен също като Universal Resource Locator (URL) или уеб адрес. URL адресът е вид единен идентификатор на ресурси (URI). В обичайната практика терминът URI не се използва или се използва синоним на URL, въпреки че това е технически неправилно. Тим Бърнърс-Ли и работната група на Internet Engineering Task Force са кредитирани за разработването на URL адреса през 1994 г. Официално е уточнено в RFC 1738.Всички URL адреси са представени в следния ред:

* + - Име на схемата
    - Двоеточие и две наклонени черти
    - Местоположение на сървъра
    - Портът (незадължително) и местоположението на ресурса на сървъра
    - Идентификатор на фрагмент (незадължително)
    - Така че форматът ще изглежда така:
* Схема: // местоположение: порт / файл-он-server.htm querystring = 1

Това изглежда по-сложно, отколкото е. Най-често срещаните схеми (протоколи) са HTTP и HTTPS, които всеки потребител на WWW ще разпознае.

Местоположението на сървъра обикновено е име на домейн. Като се има предвид това, следните URL адреси са много по-прости за разбиране:

* http://www.google.com/default.html
* https://www.google.com/default.html

И двата тези URL адреса показват, че има файл с име **default.html** на сървър с адрес на **„google.com“**. Единият използва обикновен HTTP, докато другият използва защитена версия на тази схема. Или по просто браузърът разделя URL адреса на три части:

1. Протоколът ("http")

2. Името на сървъра („www.howstuffworks.com“)

3. Името на файла ("web-server.htm")

Два общи елемента на объркване относно URL адресите:„Www“ всъщност не е част от техническия протокол. Уебсайтовете току-що започнаха да използват това, за да посочат, че потребителят използва World Wide Web. Ето защо, ако отидете на http://google.com, той се пренасочва към http://www.google.com.Повечето потребители имат достъп до Интернет чрез уеб браузър, който вмъква порт 80 на HTTP връзки зад кулисите. Ето защо, ако отидете на http://www.google.com:80, ще видите същия уебсайт, сякаш няма номер на порт. Накрая, следният URL адрес демонстрира идентификатор на фрагмент, по-известен като тест за заявка:

http://www.google.com/some-page?search=hello

Това означава, че да използвате HTTP протокола, за да изпратите заявка до уебсайта на google.com (над порт 80) и да поискате „някаква страница“ и да изпратите в променливата за търсене „здравей“. Ето защо понякога ще видите изключително дълъг URL адрес, тъй като много променливи се изпращат към уеб сървъра в по-интерактивни уеб приложения

## HTTP

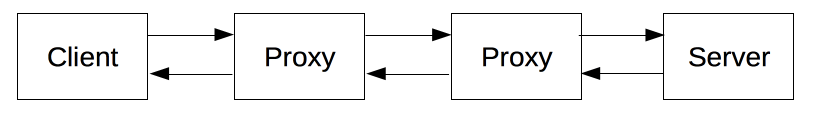
Както подсказва името му, **HTTP** ( **Hyper** **Text Transfer** **Protocol** ) указва как да се прехвърля хипертекст (свързани уеб документи) между два компютъра.Протоколът е набор от правила за комуникация между два компютъра. HTTP е текстов протокол без състояние.Нито сървърът, нито клиентът помнят предишни комуникации. Например, разчитайки само на HTTP, сървърът не може да запомни въведената от вас парола или да запомни напредъка ви при незавършена транзакция. Имате нужда от сървър за приложения за такива задачи. HTTP предоставя ясни правила за това как клиент и сървър комуникират. Обикновено само клиенти правят HTTP заявки и само към сървъри . Сървърите отговарят на HTTP заявка на клиента . Сървърът може също така да попълни данни в клиентски кеш, преди да бъде поискан, чрез механизъм, наречен [сървърно натискане](https://en.wikipedia.org/wiki/HTTP/2_Server_Push).Когато заявяват файл чрез HTTP, клиентите трябва да предоставят [URL адреса](https://developer.mozilla.org/en-US/docs/Glossary/URL) на файла.Уеб сървърът трябва да отговаря на всяка HTTP заявка, поне със съобщение за грешка.HTTP сървърът е отговорен за обработката и отговарянето на входящи заявки.При получаване на заявка, HTTP сървър проверява дали заявеният URL адрес съвпада със съществуващ файл.Ако е така, уеб сървърът изпраща съдържанието на файла обратно към браузъра. Ако не, сървърът ще провери дали трябва да генерира динамично файл за заявката.Ако нито една от тези опции не е възможна, уеб сървърът връща съобщение за грешка на браузъра, най-често [404 Not Found](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/404). Грешката 404 е толкова често срещана, че някои уеб дизайнери отделят значително време и усилия за проектиране на страници с грешка 404.

Уеб сървърите, HTTP и HTML (обяснил съм подробно на стр.10) са тясно свързани компоненти на World Wide Web. Когато уеб браузър поиска съдържание от уеб сървър, той изпраща HTTP заявка, която указва вида съдържание, което иска, и други параметри. Уеб сървърът отговаря с HTTP отговор, който включва исканото съдържание и други метаданни.HTML е езикът за маркиране, използван за създаване на уеб страници. HTML се използва за структуриране и форматиране на съдържанието на уеб страници, като текст, изображения и връзки. Когато уеб сървър получи HTTP заявка за уеб страница, той обикновено извлича HTML файл от своята файлова система и го изпраща обратно на искащия клиент като част от HTTP отговора. След това уеб браузърът използва HTML, за да визуализира страницата и да я покаже на потребителя.

Тъй като първото нещо, което прави браузърът е да изпрати HTTP заявка сега ще разгледаме какво е HTTP заявка. Основният протокол, чрез който се получават заявките и се изпраща съдържанието, се нарича HTTP заявка, а основният формат на уеб страниците е HTML. HTTP е протокол за извличане на ресурси като HTML документи. Това е основата на всеки обмен на данни в мрежата и е протокол „клиент-сървър“, което означава, че заявките се инициират от получателя, обикновено уеб браузъра. Пълният документ се възстановява от различните извлечени под документи, например текст, описание на оформлението, изображения, видеоклипове, скриптове и други. Клиентите и сървърите комуникират чрез обмен на отделни съобщения (за разлика от поток от данни). Съобщенията, изпратени от клиента, обикновено уеб браузър, се наричат ​​заявки , а съобщенията, изпратени от сървъра като отговор, се наричат ​​отговори .

Проектиран в началото на 90-те години, HTTP е разширяем протокол, който се развива с течение на времето. Благодарение на разширяемостта си, той се използва не само за извличане на хипертекстови документи, но и за изображения и видеоклипове или за публикуване на съдържание на сървъри, като с резултати от HTML формуляр. HTTP може също да се използва за извличане на части от документи, за актуализиране на уеб страници при поискване.

HTTP е протокол клиент-сървър: заявките се изпращат от един обект, потребителски агент. През повечето време потребителският агент е уеб браузър, но може да бъде всичко, например робот, който обхожда мрежата, за да попълни и поддържа индекс на търсачката.Всяка отделна заявка се изпраща до сървър, който я обработва и предоставя отговор, наречен отговор . Между клиента и сървъра има множество обекти, наричани заедно [прокси сървъри](https://developer.mozilla.org/en-US/docs/Glossary/Proxy_server) , които изпълняват различни операции и действат като шлюзове или [кешове](https://developer.mozilla.org/en-US/docs/Glossary/Cache).



В действителност има повече компютри между браузъра и сървъра, които обработват заявките: има рутери, модеми и други. Благодарение на слоестия дизайн на мрежата, те са скрити в мрежовия и транспортния слой. HTTP е отгоре, на приложния слой.

Въпреки че са важни за диагностицирането на мрежови проблеми, основните слоеве в повечето случаи са без значение за описанието на HTTP.

## [Клиент (потребителският агент](https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview#client_the_user-agent))

Потребителският агент е всеки инструмент, който действа от името на потребителя. Тази роля се изпълнява основно от уеб браузъра, но може да се изпълнява и от програми, използвани от инженери и уеб разработчици за отстраняване на грешки в техните приложения.Браузърът винаги е субектът, който инициира заявката, той никога не е сървър. За да покаже уеб страница, браузърът изпраща оригинална заявка за извличане на HTML документа, който представлява страницата. След това анализира този файл, като прави допълнителни заявки, съответстващи на скриптове за изпълнение, информация за оформлението (CSS) за показване и под ресурси, съдържащи се в страницата (обикновено изображения и видеоклипове).

След това уеб браузърът комбинира тези ресурси, за да представи пълния документ, уеб страницата. Скриптовете, изпълнявани от браузъра, могат да извличат повече ресурси в по-късни фази и браузърът съответно актуализира уеб страницата.Уеб страницата е хипертекстов документ. Това означава, че някои части от показаното съдържание са връзки, които могат да бъдат активирани (обикновено с едно щракване на мишката), за да извлекат нова уеб страница, което позволява на потребителя да насочва своя потребителски агент и да навигира в мрежата. Браузърът превежда тези указания в HTTP заявки и допълнително интерпретира HTTP отговорите, за да представи на потребителя ясен отговор.От другата страна на комуникационния канал е сървърът, който обслужва заявения от клиента документ. Между уеб браузъра и сървъра множество компютри и машини предават HTTP съобщенията. Поради слоестата структура на уеб стека, повечето от тях работят на транспортно, мрежово или физическо ниво, като стават прозрачни на HTTP слоя и потенциално оказват значително влияние върху производителността. Тези, които работят на приложните слоеве, обикновено се наричат ​​проксита . Те могат да бъдат прозрачни, препращайки заявките, които получават, без да ги променят по никакъв начин, или непрозрачни, в който случай те ще променят заявката по някакъв начин, преди да я предадат на сървъра. Прокситата могат да изпълняват множество функции:

* каширане (кешът може да бъде публичен или частен, като кеша на браузъра)
* филтриране (като антивирусно сканиране или родителски контрол)
* балансиране на натоварването (за да се позволи на множество сървъри да обслужват различни заявки)
* удостоверяване (за контрол на достъпа до различни ресурси)
* регистриране (позволяващо съхраняване на историческа информация)

## [Функциите на HTTP](https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview#basic_aspects_of_http)

HTTP обикновено е проектиран да бъде прост и четим от хора, дори с добавената сложност, въведена в HTTP/2 чрез капсулиране на HTTP съобщения в рамки. HTTP съобщенията могат да се четат и разбират от хора, осигурявайки по-лесно тестване за разработчиците и намалена сложност за новодошлите.Въведени в HTTP/1.0, [HTTP заглавките](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers) правят този протокол лесен за разширяване и експериментиране. Нова функционалност може дори да бъде въведена чрез просто споразумение между клиент и сървър относно семантиката на нова заглавка.HTTP е без състояние: няма връзка между две заявки, изпълнявани последователно на една и съща връзка. Това незабавно има изгледи да бъде проблематично за потребителите, които се опитват да взаимодействат с определени страници съгласувано, например, като използват кошници за пазаруване в електронната търговия. Но докато самото ядро ​​на HTTP е без състояние, HTTP бисквитките позволяват използването на сесии със състояние. Използвайки разширяемостта на заглавката, HTTP бисквитките се добавят към работния процес, позволявайки създаване на сесия при всяка HTTP заявка за споделяне на същия контекст или същото състояние.

## [HTTP и връзки](https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview#http_and_connections)

Връзката се контролира на транспортния слой и следователно е извън обхвата на HTTP. HTTP не изисква базовият транспортен протокол да е базиран на връзка; той изисква само да бъде надежден или да не губи съобщения (най-малкото, представяйки грешка в такъв случаи). Сред двата най-често срещани транспортни протокола в Интернет TCP е надежден, а UDP не. Следователно HTTP разчита на TCP стандарта, който е базиран на връзка.Преди клиент и сървър да могат да обменят двойка HTTP заявка/отговор, те трябва да установят TCP връзка, процес, който изисква няколко двупосочни посещения. Поведението по подразбиране на HTTP/1.0 е да отваря отделна TCP връзка за всяка двойка HTTP заявка/отговор. Това е по-малко ефективно от споделянето на една TCP връзка, когато множество заявки се изпращат в близка последователност**.** В момента се провеждат експерименти за проектиране на по-добър транспортен протокол, по-подходящ за HTTP. Например Google експериментира с [QUIC](https://en.wikipedia.org/wiki/QUIC) , който се основава на UDP, за да осигури по-надежден и ефективен транспортен протокол.Въпреки че HTTP/2 добавя известна сложност чрез вграждане на HTTP съобщения в рамки за подобряване на производителността, основната структура на съобщенията остава същата след HTTP/1.0.

Потокът на сесията остава прост, позволявайки да бъде проучен и отстранен с помощта на обикновен [HTTP монитор за съобщения](https://firefox-source-docs.mozilla.org/devtools-user/network_monitor/index.html) .HTTP е разширяем протокол, който е лесен за използване. Структурата клиент-сървър, комбинирана с възможността за добавяне на заглавки, позволява на HTTP да напредва заедно с разширените възможности на мрежата.

HTTP е протоколът, използван от уеб сървърите и уеб браузърите за комуникация помежду си. В обобщение, уеб сървърите хостват и обслужват уеб съдържание, HTTP е протоколът, използван за комуникация между уеб сървъри и клиенти, а HTML е езикът за маркиране, използван за създаване на уеб страници. Заедно тези компоненти формират гръбнака на световната мрежа. Нека видим какво е HTML, защото без него HTTP няма на какъв език да чете документите.

## HTML

Когато уеб браузър поиска уеб страница от уеб сървър, той изпраща HTTP заявка, която указва вида съдържание, което иска, и други параметри. Уеб сървърът отговаря с HTTP отговор, който включва исканото съдържание и други метаданни. Това съдържание обикновено е HTML файл, който съдържа маркирането и съдържанието на уеб страницата. HTML файлът, изпратен от уеб сървъра, се анализира от уеб браузъра, който използва маркирането и съдържанието, за да изобрази страницата и да я покаже на потребителя. HTML файлът може също така да включва препратки към други ресурси като изображения, таблици със стилове и скриптове, които уеб браузърът ще поиска от уеб сървъра с помощта на допълнителни HTTP заявки. HTML е съкращение от Hyper Text Markup Language (буквално „хипертекстов език за маркиране“). Създаден е в края на 80-те години от англичанина Тим Бърнърс-Лий, който се опитва да намери нов метод (World Wide Web) за обмяна на информация с колегите си – учени от Европейската Лаборатория по Физика на Елементарните Частици в Женева.

Същността на неговия метод е създаване на текстови документи, обвързани помежду си с хипервръзки, които да се прехвърлят чрез мрежата до отдалечен потребител. Така се появява необходимостта от специален език, на който да се пише (маркира) хипертекст. В HTML се използват маркери (тагове), които уеб браузърът чете, за да подреди съдържанието и изгради уеб страницата на екрана на потребителя. Основните елементи на HTML документа са html, head, title, body.

За да извлече уеб страница, вашият браузър изпраща заявка до уеб сървъра, който търси искания файл в собственото си място за съхранение. След като намери файла, сървърът го чете, обработва го според нуждите и го изпраща на браузъра. Нека разгледаме тези стъпки по-подробно. Първо, уеб сървърът трябва да съхранява файловете на уебсайта, а именно всички HTML документи и свързаните с тях активи, включително изображения, CSS таблици със стилове, JavaScript файлове, шрифтове и видео. Технически можете да хоствате всички тези файлове на собствения си компютър, но е много по-удобно да съхранявате всички файлове на специален уеб сървър, защото специалният уеб сървър обикновено е по-достъпен (работещ). С изключение на престой и системни проблеми, специален уеб сървър винаги е свързан с интернет. Специален уеб сървър може да има един и същ IP адрес през цялото време. Това е известно като специален IP адрес . (Не всички интернет доставчици предоставят фиксиран IP адрес за домашни цели).

Поради всички тези причини намирането на добър хостинг доставчик е ключова част от изграждането на вашия уебсайт. Разгледайте различните услуги, които компаниите предлагат. Изберете такъв, който отговаря на вашите нужди и бюджет. (Услугите варират от безплатни до хиляди лева на месец). След като имате уеб хостинг услуга, трябва да качите вашите файлове на вашия уеб сървър. Tъй като цялата тази комуникация е невъзможна без да има IP адреса. Нека да видим какво представлява той.

## IP адрес

Internet Protocol (IP) адресът е цифровият етикет, който се дава на всяко устройство, свързано в мрежа. По същество, представлява уникален номер, който се използва от машини, за да се свързват едни с други, когато изпращат информация през Интернет или локална мрежа.

Този адрес позволява на машината, която изпраща информацията, да намери с точност сървъра получател. IP адресите се различават според своето предназначение и ситуациите, в които ги използваме.

Това са виртуални адреси, които служат за комуникация в Интернет. Обикновено на всеки краен потребител се предоставя един такъв адрес, който се конфигурира на рутера му. Такива адреси имат и сървърите, които управляват потребителските заявки и насочват информацията. По това дали се променят във времето се делят на 2 вида: статични и динамични. Статичните не се променят, докато динамичните се променят през определен интервал от време. Освен тези два вида съществуват и локални IP адреси, които са ни най-добре познати, защото към тях свързваме устройствата у дома, които използваме най-често: Локални IP адреси – името им подсказва, че това са IP адреси, използвани само в рамките на дадена локална мрежа. Крайните потребители в много от случаите използват именно такива адреси, защото няма възможност да се даде публичен IP адрес на всяко едно устройство. За да може да се достъпи интернет от дома, локалните адреси използват NAT. Локални IP адреси се дават на принтери, лаптопи, смартфони, таблети.

Какво е NAT (Network address translation)? - Това е механизъм, който преобразува вътрешните адреси (private address) към външни (public address) адреси. Този механизъм се налага заради изчерпването на IPv4 адресите. Към всеки краен клиент се предоставя по един публичен адрес - най-често това е на неговия рутер, а всички други устройства, които той използва, за да достъпва Интернет трябва да преминат през механизма, за да бъде “маскиран” самият адрес. Преди няколко години, когато ставаше въпрос за IP адрес, се подразбираше, че говорим за версия IPv4 без дори да има нужда от пояснение. Сега обаче нещата са съвсем различни. IPv4 протоколът е на практика изчерпан и това води до създаването на IPv6. Нека разгледаме и основните разлики между тях.

**IPv4 адреси**

При IPv4 IP адресите са съставени от 32 бита, разделени в 4 октета с по 8 бита във всеки октет, което прави теоретично 4 294 967 296 уникални адреси. IPv4 адресите обикновено се отбелязват като четворка числа, разделени с точки. Ще дадем за пример следния публичен IPv4 адрес - 79.98.104.3. Първите три групи от числа показват мрежата, а последните цифри - адреса на хоста в мрежата.

**IPv6 адреси**

Тук вече става въпрос за Интернет протокол от следващо поколение, разработван в продължение на близо две десетилетия. IPv6 се появява в следствие на предизвикателствата, пред които ни изправя изчерпването на адресите, както и за да се справи с проблемите и недостатъците в Интернет протокола IPv4. Основен проблем при по-стария протокол е ограничението в броя на свободните IP адреси, породено от тяхната дължина (едва 32 бита). Нарастването на броя устройства, свързани към Интернет, и развитието на IoT (Internet of Things) правят необходимо наличието на много по-всеобхватно решение. IPv6 протоколът използва 128 битови адреси и има възможност за 340 трилиона възможни IP адреси. А ползите далеч не свършват дотук.

С навлизането на IPv6 всички устройства ще са достъпни от публичните мрежи, помагайки на хората лесно и бързо да управляват домашните си устройства (пускане на осветлението, отваряне на гаражната врата, управление на електроуреди и т.н.), споделянето на файлове, онлайн игри, програми и други приложения и то без сложни настройки на рутера. Версия 6 на протокола е и доста по-сигурна от старата 4. Достоверността и целостта на всеки IPv6 пакет е подсигурена от криптация посредством IP.

Новата версия на протокола е по-добра и в още нещо – достигането на Интернет трафика до правилната дестинация без той да бъде прекъсван. Браузърът комуникира със сървър за имена, за да преведе името на сървъра „www.howstuffworks.com“ в IP адрес, който използва за свързване към сървърна машина.

**Статични IP адреси**

Статичният IP адрес е фиксиран IP адрес, присвоен на устройството ръчно. Това беше първоначалният метод за присвояване на IP адреси на устройствата в мрежата. Тук за всеки компютър мрежовият администратор трябва да отвори страницата за мрежова конфигурация и ръчно да въведе IP адреса. Освен това, администраторът трябва да включва и детайли като маска на подмрежата и шлюз по подразбиране. Освен това този процес трябваше да се следва на всички устройства в мрежата. По този начин е трудно да се зададат статични IP адреси, когато има голям брой устройства в мрежата статичният IP адрес е фиксиран адрес, присвоен на устройство ръчно от мрежовия администратор, докато динамичният IP адрес е адрес, присвоен автоматично на устройство от DHCP сървър. Въпреки това, статичният IP адрес не се променя, докато не бъде ръчно променен от мрежовия администратор или доставчика на интернет услуги. Освен това, този адрес не се променя всеки път, когато потребителят се свърже с мрежата. С други думи, устройството винаги използва същия IP адрес, за да се свърже с интернет. Следователно статичното IP адресиране предоставя множество предимства. Той дава минимален престой, като осигурява отдалечен достъп. Също така, потребителят може да има достъп до устройството от всяко място.

**Динамични IP адреси**

Динамичният IP адрес е адрес, получен от DHCP сървър. Тя присвоява устройство с динамичен IP адрес, маска, шлюз по подразбиране и DNS сървър. В компютър на Microsoft, изборът на опцията „автоматично получаване на IP адрес“ в прозореца на свойствата на мрежата ще настрои устройството да получава динамичен IP адрес. Динамичният IP адрес често се променя. Всеки път, когато устройството се свърже с мрежата, динамичният IP адрес се променя. Когато устройството се опитва да се свърже с мрежата, DHCP сървърът осигурява динамичен адрес.

Статичен IP адрес или Статичен интернет протокол Адрес е постоянен цифров адрес, зададен ръчно на устройство в мрежата. Динамичният IP адрес или Адресът на динамичния интернет протокол е временен IP адрес, който е присвоен на устройство или възел, когато е свързан към мрежа. По този начин, това е основната разлика между статичен и динамичен IP адрес. Друга разлика между статичния и динамичния IP адрес е, че статичният IP адрес се задава ръчно от мрежовия администратор, докато динамичният IP адрес се задава автоматично от DHCP сървъра. Статичният IP адрес не се променя, след като е присвоен на устройство.

Въпреки това динамичният IP адрес се променя всеки път, когато устройството се свързва с мрежата. Следователно, това е и основна разлика между статичния и динамичния IP адрес. Също така, задаването на статични IP адреси е трудно. Но присвояването на динамични IP адреси е по-лесно. Що се отнася до използването, статичните IP адреси са подходящи за специализирани услуги като поща, FTP и VPN сървъри. Динамичните IP адреси са подходящи за голяма мрежа, която изисква достъп до интернет за всички устройства. заключение IP адресът помага на устройствата в мрежата да комуникират с други устройства.

Най- основната разлика между статичен и динамичен IP адрес е, че статичният IP адрес е фиксиран адрес, присвоен на устройство ръчно от мрежовия администратор, докато динамичният IP адрес е адрес, присвоен автоматично на устройство от DHCP сървър.

## Какво е DHCP и DHCP сървър?

DHCP – Dynamic Host Configuration Protocol е протокол за управление на мрежата, който използваме в TCP/IP мрежа. DHCP сървърът автоматично присвоява IP адреси и други мрежови конфигурации като подмрежова маска, шлюз по подразбиране, DNS сървър и други на свързаните устройства, така че те да могат да обменят информация. DHCP позволява на хостовете да получат необходимите TCP/IP конфигурационни данни от DHCP сървъра. Устройство прави заявка за IP адрес, ако иска да получи достъп до мрежа, която използва DHCP. Сървърът отговаря и предоставя IP адрес на устройството. След това той следи използването на адреса и когато изтече определен период или устройството се изключи, го връща обратно към своя пул от налични IP адреси. Той се съхранява, докато не трябва да бъде преназначен на друго устройство, което иска достъп до мрежата. Използвайки този протокол, мрежовите администратори не трябва да задават статичен IP за всяко устройство и по-късно да го преназначават на друго и да следят всички налични IP адреси. Те просто ще настроят DHCP сървъра с цялата допълнителна мрежова информация и той ще върши работата си динамично. Защо DHCP е важен? DHCP е важен, защото улеснява мрежовите администратори да предоставят IP адреси на клиентски устройства в мрежата. Той управлява автоматично пула от IP адреси. Клиентът също не трябва да прави нищо. Новосвързаното устройство автоматично ще поиска IP адрес и ще го получи. Човекът зад устройството не трябва да прави никакви конфигурации. Всяко устройство в мрежата се нуждае от адреса като идентификация. И две устройства не могат да имат едно и също IP, защото това ще ги направи и двете неизползваеми.

Компоненти на DHCP сървър . Сървърното устройство отговаря за отговора на заявка за IP адрес, предоставя наличен IP адрес, съхранява го за времето на лизинга и го подновява по-късно. Той ще управлява комуникацията с всички клиентски устройства. Сървърът може да бъде компютър или част от рутера.

DHCP клиент. Той трябва да присъства на клиентските устройства (компютър, мобилно устройство, IoT устройство и др.). Той ще поиска IP адрес и ще комуникира с DHCP сървъра, за да го получи с останалите данни и да потвърди процеса.

DHCP обхват е диапазонът от IP адреси, които DHCP сървърът може да предложи на DHCP клиентите.

Обикновено сървърът ще задава автоматично адреси, започвайки от най-малкото число и стигайки до най-високото. Ако мрежата е разделена на части, ще има така наречените подмрежи. Отдаване под наем. Това е периодът от време, който показва колко дълго клиентът може да използва присвоения IP адрес, преди да изтече. DHCP реле отговаря за комуникацията между DHCP сървъра и клиента. Той ще слуша за съобщения и ще ги предава на правилното място. Механизми за разпределяне на IP адреси на DHCP.

Има три начина, по които можете да конфигурирате DHCP сървъра: Автоматично разпределение. То ще присвои IP на клиент за постоянно. IP адресът ще бъде определен само за едно устройство, така че ако в бъдеще се свържат много нови устройства, сървърът може да изчерпи IP адресите за предоставяне.

Динамично разпределение То е най-често срещаната конфигурация. Сървърът автоматично присвоява IP адреси на клиенти, но има период от време. След изтичане на времето клиентът трябва отново да поиска нов IP адрес. Това ще предотврати изчерпването на IP адресите. Ръчно разпределение. Ръчно мрежовият администратор ще присвои IP адреса на клиента.

Как работи DHCP? Представете си, че имаме мрежа от свързани устройства и DHCP сървър, който управлява IP адресите.

* Стъпка 1: Открийте DHCP Когато свържете ново устройство, то все още няма IP адрес. Той ще търси IP адрес. Ще се обади по мрежата за DHCP сървър. Тази заявка ще пристигне до всички устройства и сървърът също ще я получи.
* Стъпка 2 DHCP оферта DHCP чува повикването и отговаря с IP адрес, който го предлага на новосвързаното устройство.
* Стъпка 3 DHCP заявка IP адресът пристига на устройството. Устройството ще го приеме и ще изпрати заявка за използването му.
* Стъпка 4 DHCP пакет Сървърът получава приемащото съобщение от устройството. Той ще предостави IP адреса на устройството, заедно с подмрежовата маска и DNS сървъра. Той ще запише запис с информацията за новосвързаното устройство, което обикновено включва MAC адреса на свързаното устройство, присвоения IP адрес и датата на изтичане на този IP адрес.

DHCP наема IP адреса само за ограничен период от време. След като изтече времето, IP адресът ще се върне обратно към IP пула от налични IP адреси и може отново да бъде присвоен на ново устройство. UDP портът за комуникации обикновено е порт 68 за клиенти и порт 67 за сървъри. Възможно е да има някои разлики в зависимост от доставчиците на мрежово оборудване, но това е как функционира като цяло.

## Каква е основната функция на уеб сървърите?

Основната функция на уеб сървъра е да съхранява съдържанието на уеб страниците, уеб приложения и други, да получава и обработва заявки, идващи от клиенти (браузъри) и да отговаря на тези заявки, изпращайки съответното съдържание към клиента.С развитието на технологиите, съдържанието на уеб страниците става все по-разнообразно и с това се разширяват функциите и възможностите на уеб сървърите.Ако някога функциите им се свеждаха до това да предоставят текст и изображения при поискване, днес се използват за съхранение и обслужване на всякакъв вид сложни приложения, като уеб услуги, контролни панели, социални медии и други.

## DNS

DNS системата се използва в интернет [от 1985 г.](https://en.wikipedia.org/wiki/Domain_Name_System), ~5 години преди появата на Уеб. Първата реализация на DNS системата е била за UNIX и е написана от четирима студенти от университета Berkeley. Наречена е The Berkeley Internet Name Domain – [BIND](https://bg.wikipedia.org/wiki/BIND). В днешни дни BIND е най-разпространеният софтуер за DNS сървър, който може да се ползва и за DNS резолвер, и за отговорен нейм сървър.DNS ([Domain Name System](https://en.wikipedia.org/wiki/Domain_Name_System)) е система за имена в интернет, чрез която домейните се транслират в IP адреси. Видимата част на DNS системата са домейните, а целта им е да се предостави начин за наименуване на ресурсите/услугите в интернет.

Почти всички услуги в интернет ползват DNS като [уеб](https://help.superhosting.bg/world-wide-web.html), имейл услугата, услугите за трансфер на файлове и други. Благодарение на DNS отваряме уеб сайтовете в уеб браузъра, като само изпишем името на домейна им. Без тази система, вместо домейн ще трябва да въвеждаме IP адреса на сървъра, на който реално се намира съдържанието на сайта. Защо домейните трябва да се транслират в IP адреси? Лично на нас, хората, IP адресът не ни е нужен. Много по-лесно е да изпишем и запомним домейна „superhosting.bg“, вместо IP адреса на сървъра му „91.196.125.21“. **IP адресът е нужен на приложението, в случая уеб браузъра, за да открие сървъра в интернет, на който се намира уеб сайта, и да се свърже с него.** Най-общо казано, за да могат две приложения да комуникират помежду си в интернет, едното задължително трябва да знае IP адреса на другото. Това е основна характеристика на комуникацията [клиент-сървър](https://en.wikipedia.org/wiki/Client%E2%80%93server_model) в интернет. **DNS системата се използва като виртуален Указател, в който са вписани**[**(DNS) записите**](https://blog.superhosting.bg/dns-zone-how-the-dns-works.html)**„домейн име= IP адрес“.** Всеки, който поиска да узнае IP адреса на даден домейн, ще трябва да потърси в DNS указателя важния DNS запис „домейн име= IP адрес“.От DNS указателя имат нужда всички приложения, които трябва да се свържат с даден сървър в интернет, но разполагат само с домейн име. Например уеб браузърът трябва да разбере какъв е IP адреса на сървъра, на който се намират файловете на сайта mysupersite.com, мейл клиентът има нужда да знае IP адреса на входящия мейл сървър, който сме въвели като домейн mail.mysupersite.com. FTP клиентът също ще иска да разбере IP адреса на FTP сървъра, който сме въвели като домейн“ [ftp.mysupersite.com](ftp://ftp.mysupersite.com).“

Пространство от имена и ресурсни записи (RR)

Търсеният запис се намира в един от [основните три компонента на DNS системата](https://datatracker.ietf.org/doc/html/rfc1034).Пространството от имена има дървовидна структура, а всяко ниво от него се нарича домейн име. Ресурсните записи (RR) са информация, която е асоциирана с дадено домейн име. Ресурсните записи се наричат още [DNS записи](https://blog.superhosting.bg/dns-zone-how-the-dns-works.html) и се съхраняват в обикновен текстов файл, наричан DNS зонов файл. Важният запис „домейн име=IP адрес“ е DNS запис от тип „А“. Нейм сървърът съхранява DNS зоната за даденото домейн име. Важният DNS запис „домейн име=IP адрес“ се намира в текстов файл, разположен на нейм сървър. Този нейм сървър, на който се съхранява „достоверната“ DNS информация за домейна, се нарича [Authoritative нейм сървър](https://en.wikipedia.org/wiki/Name_server#Authoritative_name_server). Този нейм сървър отговаря за DNS записите на този домейн. Authoritative нейм сървърите на български език може да се нарекат – отговорни или обслужващи нейм сървъри.

Ще ги наричаме отговорни нейм сървъри, за да ги различаваме от DNS кеша и DNS резолвера.Всички други DNS сървъри в интернет, които се допитват до „отговорния“ нейм сървър, за да получат информация за даден домейн, са DNS резолвери.

DNS резолверът извършва проверка в указателя, като следва препратките от едно ниво към следващо, от един отговорен нейм сървър към друг, докато не стигне до търсения запис.Резолверът е приложение, най-често намиращо се на DNS сървър при интернет доставчика. Към него може да работи още едно приложение, кеш сървър, което да запазва резултатите от извършените DNS проверки.Към операционната система на десктоп компютрите и мобилните устройства има вграден локален DNS клиент, който също се нарича резолвер (stub-resolver). Уеб браузърът си комуникира с локалния DNS клиент, който от своя страна се обръща за съдействие към резолвера на интернет доставчика.

Запитване до DNS указателя за IP адреса на домейна

Откриването на IP адреса изисква приложението-клиент да зададе специфични DNS въпроси (по DNS протокола) към DNS указателя. Уеб браузърът не знае тези DNS въпроси, но знае един специален въпрос: „Кой е IP адресът на този домейн?“ и знае точно на кого да го зададе – на локалния DNS клиент. А в отговор ще очаква само и единствено IP адрес. **Тъй като уеб браузърът не е предназначен да извършва DNS проверки в DNS указателя, той ще се свърже с друго помощно приложение, което ще открие IP адреса на домейна вместо него. Това е локалният DNS клиент, който ще се** свърже с DNS резолвера, намиращ се на DNS сървър при интернет доставчика. Ако използвате Windows (7/8/10) можете да видите кои са DNS резолверите, към които локалния резолвер ще се обърне за съдействие с намирането на IP адреса на домейна. (Отворете с десен бутон върху иконката на мрежата » Open Network and Sharing Center » Change adapter settings, десен бутон на мишката върху активната връзка » Properties, маркирайте Internet Protocol Version 4 (TCP/IPv4) и кликнете Properties. В долната част на прозореца ще видите настройките за DNS сървърите, които ще се ползват от вашата система, при всяка нужда от проверка в указателя. Посочените на изображението DNS резолвери са на [Google Public DNS](https://developers.google.com/speed/public-dns)). След като локалният резолвер (локален DNS клиент) е получил въпроса на браузъра, той първо ще провери в [DNS кеша на системата](https://help.superhosting.bg/clear-local-dns-cache.html) и в [хост файла](https://help.superhosting.bg/hosts-file.html), дали случайно в тях няма информация за този домейн, и ако няма, тогава ще изпрати запитване за съдействие към **DNS резолвера**.След като резолверът получи запитване за IP адреса на домейна, той ще стартира проверка в указателя.Резолверът е приложение, което приема заявки от DNS клиенти, разпитва един или повече отговорни нейм сървъри в указателя и връща на клиента отговор с търсената информация, за да може да върне на клиента IP адрес, резолверът ще търси „А“ записа, в който се съдържа IP адреса на домейна („домейн име=IP адрес).Тук все още няма да включваме DNS кеш приложенията, чиято единствена цел е да записват резултатите от DNS проверките, така че при следващо клиентско запитване за същата информация, да спестят на резолвера и отговорните нейм сървъри същата DNS проверка.В текстовия файл, в който се съхранява „А“ записът, са изброени в списък и всички други DNS записи за дадения домейн.

Този списък със записи се нарича DNS зона.За да стигне до отговорния нейм сървър, на който се съхранява DNS зоната на домейна, DNS резолверът ще премине през всички нива в DNS указателя, като всяко ниво ще го препраща към следващото, и така докато не намери крайната цел. А DNS проверката ще започне от основното ниво на DNS..

## Какво се случва при зареждането на даден уебсайт?

За да достъпите даден уеб сайт в уеб пространството, е необходимо да имате инсталирано специално приложение на компютъра, например уеб браузър.Когато името на сайта е въведено, за да зареди някакво съдържание уеб браузърът първо трябва да открие на кой сървър се намират уеб страниците на този сайт. Това става като извърши DNS проверка на домейн името на сайта. След като разбере кой е сървъра, уеб браузърът се свързва с инсталирания на него уеб сървър.

Контактът между двете приложения – уеб браузър и уеб сървър, се осъществява с TCP/IP протоколите за свързване в интернет. Мястото на срещата, по подразбиране, е на територията на уеб сървъра на комуникационен порт номер 80. А езикът, на който двете приложения си говорят, е протоколът за пренос на хипертекст HTTP (Hypertext Transfer Protocol).През първите години на World Wide Web всяка уеб страница се е намирала локално на компютъра на потребителя, който я е създал. За да може уеб страницата да е достъпна в уеб пространството, е трябвало на същия компютър да се инсталира и уеб сървър.В днешно време хостинг услугите ползват същата тази технология, като на всеки хостинг сървър има инсталиран уеб сървър и уеб страниците се намират на същият този хостинг сървър. Например изписваме в уеб браузър уеб адреса: help.superhosting.bg/domain-name-system.htmlКогато уеб браузърът се свърже с уеб сървъра, преди да се зареди уеб страницата се случва следното:

* уеб браузърът установява връзка с уеб сървъра на порт 80 (http);
* уеб браузърът изпраща заявка до уеб сървъра за уеб страницата, например GET domain-name-system.html;
* уеб сървърът връща отговор с HTML кода, който се намира в търсения файл;
* уеб браузърът чете този HTML код и изгражда уеб страницата на екрана.

## Уеб сървър като услуга

Като всяка програма уеб сървърите имат определени изисквания към средата, но могат да работят и на всеки наличен компютър. Когато обаче искаме даден уебсайт да бъде достъпен до широка публика без прекъсване, тогава задачата вече не е толкова проста. В много случаи уебсайтовете са важна част от даден бизнес и са източник на средства. Това поставя по-високи изисквания към тяхното [хостване](https://help.superhosting.bg/web-hosting.html).Трябва да се осигури непрекъснато електрозахранване, постоянна интернет връзка, техническа поддръжка, защита и наблюдение, за да бъдем сигурни, че съдържанието им ще бъде достъпно по всяко време.

Поради тази причина уебсайтовете и съответно уеб сървърите работят на сървърен хардуер, в специални колокационни центрове, където са осигурени необходимите условия, за да се поддържа качеството на услугата.Самите уеб сървъри, като софтуер, имат множество особености и изискват настройка и грижа. Това е задачата на хостинг компаниите – да осигурят оптимални условия за съхранение и обслужване на уеб съдържанието.Ако става въпрос за комерсиален сайт с хиляди посещения на ден, неговото поддържане е невъзможно в домашни условия на собствен компютър. Ето защо сайтовете масово се съхраняват и обслужват от хостинг компании, които разполагат с нужния опит и необходимите условия за целта.

## Коя операционна система да изберем за web сървъра ни.

Има няколко операционни системи, които обикновено се използват за уеб сървъри, включително Windows, Linux и macOS. Всеки има своите предимства и недостатъци и изборът на операционна система в крайна сметка зависи от редица фактори, включително вашите специфични нужди и предпочитания. Linux често се смята за най-популярния избор за уеб сървъри поради своята стабилност, сигурност и естество с отворен код. Той е много персонализиран и може лесно да се конфигурира, за да отговори на специфичните нужди на уеб сървърите.

Освен това, много популярен софтуер за уеб сървъри, като Apache и Nginx, са проектирани да работят на Linux. Windows е друг популярен избор за уеб сървъри, особено за организации, които използват технологии на Microsoft. Windows Server е мощна операционна система, която предлага широк набор от функции и инструменти за управление на уеб сървър. Освен това, много популярен софтуер за уеб сървъри, като IIS (Интернет информационни услуги), е проектиран да работи под Windows. macOS също е жизнеспособна опция за уеб сървъри, особено за внедрявания в по-малък мащаб или за разработчици, които предпочитат базирана на Unix операционна система. macOS е стабилна и лесна за използване операционна система, която включва много мощни инструменти за управление на уеб сървъри, като Apache и Nginx. В обобщение, изборът на операционна система за уеб сървър в крайна сметка зависи от вашите специфични нужди и предпочитания. macOS също е жизнеспособна опция за внедрявания в по-малък мащаб или за разработчици, които предпочитат базирана на Unix операционна система. Сървърните операционни системи са предназначени за платформи, които действат като уеб сървъри, сървъри на приложения, [сървъри на бази данни](https://phoenixnap.com/kb/what-is-a-database-server) , сървъри за електронна поща или други платформи, работещи на специален сървър.Сървърните операционни системи предлагат различни предимства в сравнение със стандартните [операционни системи](https://phoenixnap.com/glossary/operating-system) , включително неограничени потребителски връзки, повече памет и разширени мрежови оптимизации.Сървърната операционна система е усъвършенствана операционна система, предназначена да работи на сървъри. Той има необходимите функции и подсистеми, за да работи в [архитектура клиент-сървър](https://phoenixnap.com/glossary/client-server) и да обслужва заявките на клиентите.Сървърната операционна система е проектирана от самото начало, за да предоставя функции, подходящи за много потребители, критични за бизнеса [приложения](https://phoenixnap.com/glossary/what-is-an-application) . Той осигурява централен интерфейс за управление на потребители, прилагане на защита и други административни процеси. Фокусът на сървърната операционна система обикновено е сигурността, стабилността и сътрудничеството.

Сървърните операционни системи улесняват внедряването на различни сървърни платформи, като например:Уеб сървъри . Уеб сървър хоства програми и данни и отговаря на клиентски заявки за уеб страници или други уеб базирани услуги. Обичайните уеб сървъри са [Apache](https://phoenixnap.com/kb/install-tomcat-windows) , Microsoft Internet Information Services (IIS) и [Nginx](https://phoenixnap.com/kb/how-to-install-nginx-on-ubuntu-20-04). Пощенският сървър позволява на потребителите да препращат и получават имейли за и от своя бизнес и контролира отделни имейл акаунти въз основа на конкретен домейн.Споделянето на файлове включва съвместна точка за съхранение за бизнес за съхраняване на документи или данни - мрежово устройство.Сървъри за бази данни някои сървърни операционни системи включват интегриране на бази данни, което улеснява изграждането на динамична уеб страница въз основа на съдържанието на базата данни. Сървъри за приложения . Сървърните операционни системи трябва да могат да изпълняват критични за бизнеса приложения, независимо дали това е самостоятелно хостван CRM или SaaS. Сървърната операционна система обикновено функционира като споделена среда за различни приложения за съвместна работа. Друга функция, която сървърните операционни системи улесняват, е споделянето на печат, което позволява на множество машини да използват един принтер.Днес има изобилие от операционни системи. Според някои статистики приблизително 80% от всички сървъри използват някакъв вариант на Linux , докато около 20% от сървърите използват Windows .

## Основните разлики между Windows и Linux.

Рязката разлика в пазарния дял вероятно е, защото Linux е безплатен и не изисква потребителско лицензиране като Windows.Всяка ОС има своите плюсове и минуси, а лекотата на използване зависи от нивото на техническа компетентност на потребителя. Следният списък не е изчерпателен, но включва най-често срещаните сървърни операционни системи.

1. **Windows сървър**

Microsoft разработи семейството операционни системи Windows за ежедневна лична употреба и професионална употреба в сървъри. Windows Server OS поддържа управление на корпоративно ниво, съхранение на данни и огромен набор от приложения. Windows Server разполага с управление [на виртуална памет](https://phoenixnap.com/glossary/virtual-memory-definition) , пълно изживяване с GUI на работния плот, позволява многозадачност и поддържа различни периферни устройства. Microsoft обикновено предоставя 10 години поддръжка за Windows Server.Предимствата на Windows server OS са интуитивният GUI, поддръжката на симетрични многопроцесорни системи, страхотната поддръжка на приложения на трети страни и много версии, от които да избирате . Минусите са необходимостта от базирано на потребителите лицензиране и повече заплахи за сигурността от вируси в сравнение с други платформи .

1. **Linux сървър**

Linux е семейство UNIX-подобни операционни системи, които имат всички характеристики на UNIX . Той е с отворен код, достъпен безплатно, улесняващ операции с много потребители, много процеси и много нишки.

Linux сървърите обаче изискват повече технически познания – от инсталацията до поддръжката и отстраняването на грешки.Една от най-разпространените платформи за хостване на уебсайтове, [стекът LAMP](https://phoenixnap.com/kb/what-is-a-lamp-stack) , е базиран на Linux (Linux, Apache, MySQL, [Perl](https://phoenixnap.com/glossary/what-is-perl) /PHP/Python).Някои от най-популярните сървърни дистрибуции на Linux са:

* Ubuntu сървър.
* Debian сървър.
* Fedora.
* OpenSUSE Leap.
* SUSE Linux Enterprise Server.
* Arch Linux.

Плюсовете на операционната система за сървъри на Linux са висока степен на сигурност, голямо разнообразие от дистрибуции, интегриран софтуер с отворен код, включително езикови компилатори на високо ниво и възможност за управление на системата с помощта на GUI.Минусите са липсата на дългосрочна поддръжка за някои дистрибуции и някои сложни операции, като системни актуализации .

1. **Red Hat Enterprise Linux (RHEL)**

RHEL е платена Linux десктоп и сървърна дистрибуция, създадена от Red Hat. Първоначално RHEL беше пуснат като Red Hat Linux Advanced Server и по-късно преименуван на Red Hat Enterprise Linux AS , който включваше две дистрибуции - Red Hat Enterprise Linux ES и Red Hat Enterprise Linux WS .Изходният код на RHEL е свободно достъпен, но Red Hat използва строги разпоредби, които ограничават преразпределението на официалната версия на Linux OS. Тези ограничения не се прилагат за производни на трети страни, които не включват несвободните компоненти, като например търговските марки на Red Hat.Плюсовете на RHEL са обширната поддръжка и наличните корекции, надстройки и решения за уязвимости в сигурността . Минусите на RHEL са скъпите курсове за обучение и липсата на персонализирани решения.

1. **UNIX-базирани операционни системи**

В началото UNIX беше операционна система за споделяне на времето за малки компютри и с течение на времето се превърна в една от най-разпространените операционни системи за среда клиент-сървър. Езикът за програмиране на UNIX е C, което улеснява създаването на UNIX портове за много машини.Предимствата на UNIX са многопотребителска среда, вградена TCP/IP поддръжка и високо ниво на стабилност и сигурност . Недостатъкът е, че се плаща и различни доставчици продават различни версии на UNIX, така че няма стандартна версия на UNIX .

1. **NetWare**

Novell NetWare е базирана на сървър [мрежова операционна система](https://phoenixnap.com/glossary/network-operating-system) , която изисква [специален сървър,](https://phoenixnap.com/servers/dedicated) за да функционира. Това беше широко разпространена операционна система в ранните локални мрежи.Плюсовете на NetWare са неговата поддръжка за мултипроцесори и управление на физическа памет с голям капацитет, както и първокласни функции за споделяне на файлове и печат в корпоративни мрежи. Операционната система също предлага широка гама от интерфейси за управление, включително уеб интерфейс.Недостатъците са цената, лошата поддръжка, предизвикателният процес на инсталиране и ниската поддръжка на приложения на трети страни .

1. **macOS сървър**

MacOS Server е UNIX-подобна сървърна операционна система, базирана на macOS, разработена от Apple. ОС се основава на macOS и добавя сървърна функционалност и инструменти за системно администриране, както и инструменти за управление на macOS и iOS устройства.MacOS Server е чудесен избор, ако използвате Mac клиенти във вашата мрежа, като се има предвид способността му лесно да създава функции за Mac клиенти.

Плюсовете на macOS Server са лесно администриране, интуитивен GUI, страхотна поддръжка и лесно разпределение на работното натоварване между множество машини. По този начин е лесно да се увеличи мощността на обработка. ОС идва с неограничен потребителски лиценз.Минусите са, че macOS Server работи само на хардуер на Apple, което може да бъде скъпо, и няма много приложения на трети страни.

Освен това, докато Apple внедрява софтуер с отворен код в своята система, има промени, специфични за macOS, изискващи заобикаляне на някои проблеми, които не присъстват в Linux.

1. **FreeBSD**

FreeBSD е безплатна Unix-подобна операционна система с отворен код. ОС поддържа цялостна система, предоставяща [ядрото](https://phoenixnap.com/kb/build-linux-kernel) , драйверите, помощните програми и документацията и включва обширна колекция от софтуер, свързан със сървъра. Така FreeBSD лесно се конфигурира като пощенски сървър, уеб сървър, защитна стена и т.н.FreeBSD има свой екип за сигурност, който инспектира целия софтуер, доставен с базовата дистрибуция, и позволява инсталирането на приложения на трети страни от двоични пакети.Плюсовете на FreeBSD са, че е бърз, напълно безплатен, има добра сигурност, която използва ipfw защитната стена и има много налични инструменти, притежавани от екипа на FreeBSD **.** Минусите на FreeBSD са, че не е много лесна за научаване, тъй като има по-слаба поддръжка от общността от Linux и липсва поддръжка на драйвери **.**

**След като разгледахме някои от най-популярните операционни системи в света. Най-популярната и използвана операционна система е Linux. А от дистрибуциите на тази операционна система е Linux Ubuntu. Нека да видим защо аз се спрях на точно тази операционна система и дистрибуция** Linux Ubuntu.

## Защо Linux Ubuntu?

Linux Ubuntu е една от най-популярните операционни системи за уеб сървъри, защото предлага няколко предимства за администриране на уеб сървъри, включително:

* Отворен код: Ubuntu е операционна система с отворен код, което означава, че е безплатна за използване, разпространение и модифициране. Това го прави привлекателна опция за администраторите на уеб сървъри, които искат да избегнат скъпи лицензионни такси.
* Стабилност и надеждност: Ubuntu е известен със своята стабилност и надеждност, което го прави отличен избор за администриране на уеб сървър. Освен това има версия за дългосрочна поддръжка (LTS), която се поддържа до 5 години, предоставяйки на потребителите стабилна и надеждна платформа за техните уеб сървъри.
* Сигурност: Ubuntu има силен фокус върху сигурността, с редовни актуализации и корекции за сигурност. Това го прави популярен избор за администраторите на уеб сървъри, които искат да гарантират сигурността на своите уеб сървъри и данните, съхранявани на тях.
* Лесен за използване: Ubuntu е проектиран да бъде лесен за използване, с удобен за потребителя интерфейс и лесен процес на инсталиране. Това го прави популярен избор както за начинаещи, така и за опитни потребители.
* Съвместимост: Ubuntu е съвместим с широк набор от софтуер и приложения, което го прави отличен избор за администриране на уеб сървър. Той също така поддържа различни програмни езици и рамки, включително PHP, Python и др.

Тези точки убедиха мен и повечето фирми и бизнеси да използват Linux Ubuntu. **Нека сравним двата най-използвани web сървъри.**

## Apache или IIS?

Кое да изберем? Както всяко едно решение и този избор зависи от няколко ключови компонента: **цена, поддръжка, възможности**. Цена и поддръжка са склонни да вървят ръка за ръка. Дори когато избирате уеб сървър с отворен код е възможно да има скрити разходи, свързани с поддръжката му. Много компании с отворен код предлагат професионален съпорт, който ще ви позволи да получите максималното от продукта ви.

Нека да разгледаме техните предимства и недостатъци. Възможностите, които предоставят отделните web платформи, може също да определят кой web сървър ще изберем. Един популярен метод е да се помисли сдвояването на операционната система и уеб сървър с рамка от база данни и програмна среда.

Има няколко подобни варианта:

* **LAMP:** Linux, Apache, MySQL и PHP
* **WIMP:** Windows, IIS, MySQL / MS SQL Server и PHP
* **WAMP:** Windows, Apache, MySQL / MS SQL Server и PHP
* **LEMP:** Linux, Nginx, MySQL и PHP

Apache и IIS имат своите плюсове и минуси. IIS трябва да бъде в пакет с Windows, но Apache не разполага с корпоративна подкрепа. Apache има отлична сигурност, но не предлага безупречната поддръжка на IIS. Както показва статистиката Apache и IIS са основните играчи на пазара.

**Apache**

**Цена:** Отворен код, безплатен, без лицензионни такси.

**Предимства:**

Гъвкав, поради способността потребителите да избират различни модули.

Предлага по-голяма сигурност, което е очаквано, поради факта, че Windows – базираните машини са най-често подложени на атаки.

Ползва се с широка подкрепа от потребителската общност.

Работи под UNIX, Windows, Linux, Mac OS

**Недостатъци:**

Apache е процесорно базиран сървър, което означава, че всяка връзка изисква проводник, който може да понесе значително натоварване.

Известни сайтове използващи Apache: Baidu, Wikipedia, PayPal, Apple, Huffington Post

**Microsoft IIS**

**Цена:** Предоставя се с лиценза за закупуване на Windows ( това означава, че цената варира в съответствие с политиката на Microsoft)

**Предимства:**

Поддържа се от Microsoft.

Предоставя достъп до .NET framework & ASPX скриптове.

IIS е интегриран с други Microsoft базирани сервизи като Active Directory, MS SQL server, ASP и прочие.

**Недостатъци:**

От гледна точка на персонализацията не е толкова гъвкав колкото open-source web платформите.

Популярни сайтове използващи IIS: Live.com, Bing.com, Microsoft, MSN, Barbie.com.

След като разгледахме предимствата и недостатъците на двата web сървъри. Стана ясно, че Apache бие по всичко Microsoft IIS.

Популярен метод е да се помисли сдвояването на операционната система и уеб сървър с рамка от база данни и програмна среда. Един от най-известният е LAMP(Linux, Apache, MySQL и PHP). Аз реших да го използвам и в нашия сървър.

## Виртуални среди и пакети

Приложенията на Python често използват пакети и модули, които не идват като част от стандартната библиотека. Приложенията понякога ще се нуждаят от конкретна версия на библиотека, тъй като приложението може да изисква определена грешка да е коригирана или приложението може да бъде написано с помощта на остаряла версия на интерфейса на библиотеката. Това означава, че може да не е възможно една инсталация на Python да отговори на изискванията на всяко приложение. Ако приложение A се нуждае от версия 1.0 на определен модул, но приложение B се нуждае от версия 2.0, тогава изискванията са в конфликт и инсталирането на версия 1.0 или 2.0 ще остави едно приложение неспособно да работи.

След това различни приложения могат да използват различни виртуални среди. За да разреши по-ранния пример на противоречиви изисквания, приложение A може да има собствена виртуална среда с инсталирана версия 1.0, докато приложение B има друга виртуална среда с версия 2.0. Ако приложение B изисква надграждане на библиотека до версия 3.0, това няма да засегне средата на приложение а.

За да бъде по лесно за разработчиците на сайтове трябва да има в сървъра за всеки сайт отделна виртуална среда. Всеки разработчик да си сваля необходимите пакети и библиотеки необходими за неговия сайт.

## Инсталиране на сървъра

Картина, която съдържа текст

Описанието е генерирано автоматичноСлед като сме избрали операционна система и нейната най-подходяща за нас дистрибуция съответно и подходящ пакет, а именно Linux Ubuntu и LAMP. Следва да я инсталирам на нашия предоставен от училището сървър. Или по-просто да инсталирам избрания софтуер на нашия хардуер. Нека започнем с направата на инсталационна флашка като си изтеглим инсталационен файл за сървър от официалния сайт на Ubuntu с най-новата версия за сървър в нашия случай 22.04 и да го качим на нашата флашка. Вземаме нашия хардуер и се закачаме с инсталационната флашка. Чакаме да зареди и започваме с инсталационен процес. С първия прозорец за избор на език:

Хубаво щеше да бъде да има и български, нo няма, затова избираме Английския. Следващия прозорец е за избиране на инсталация с ъпдейти или без тях. Аз избрах с тях:

Картина, която съдържа текст

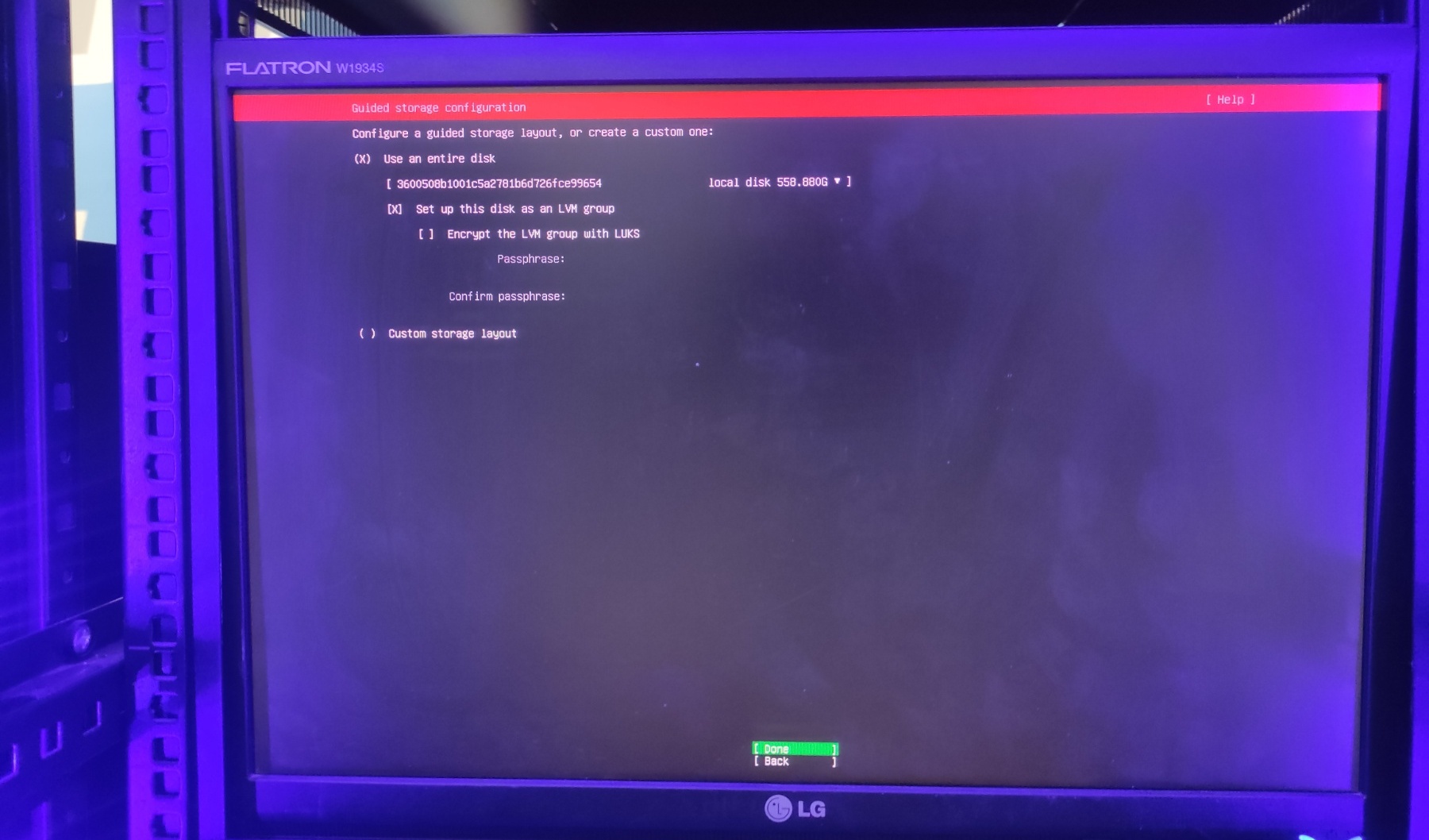
Описанието е генерирано автоматично

Следващия прозорец е за избиране на клавиатура - избираме Английски, няма да ни трябва кирилица:

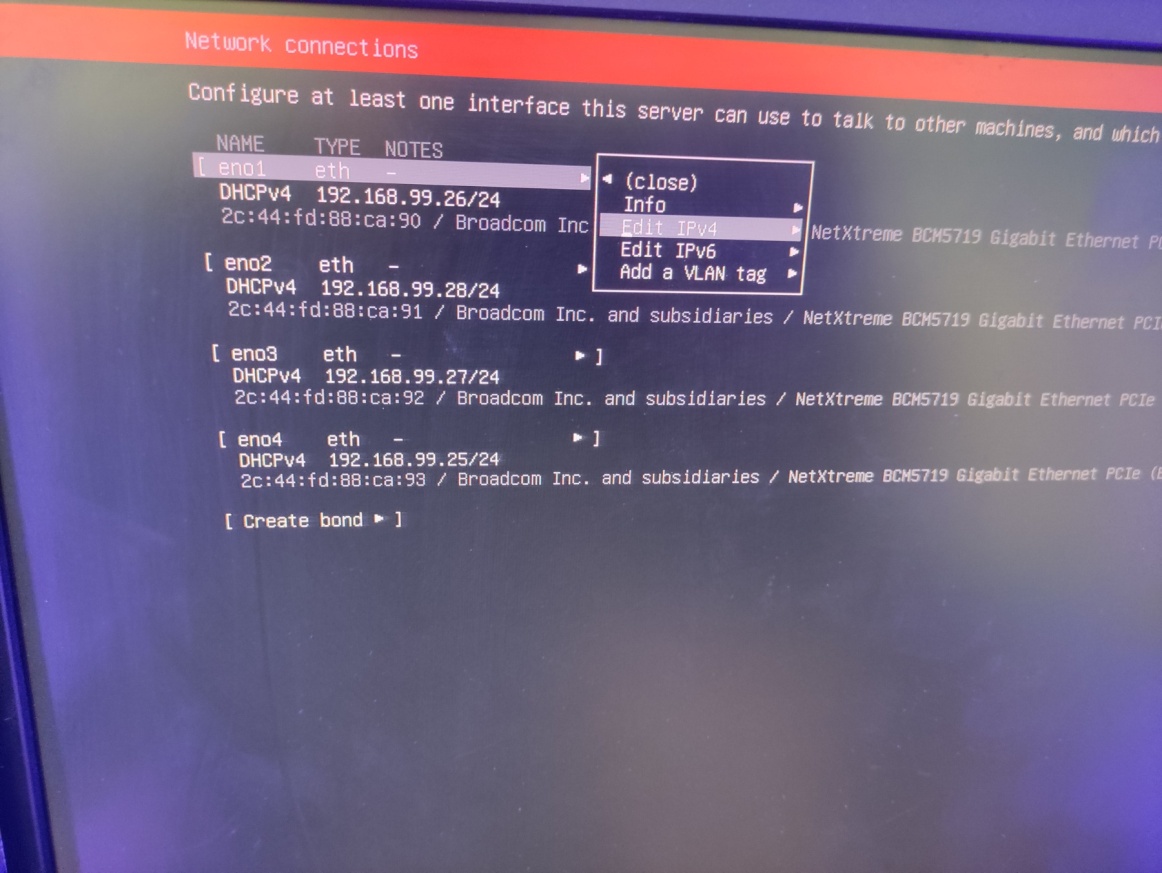
Картина, която съдържа текст

Описанието е генерирано автоматично

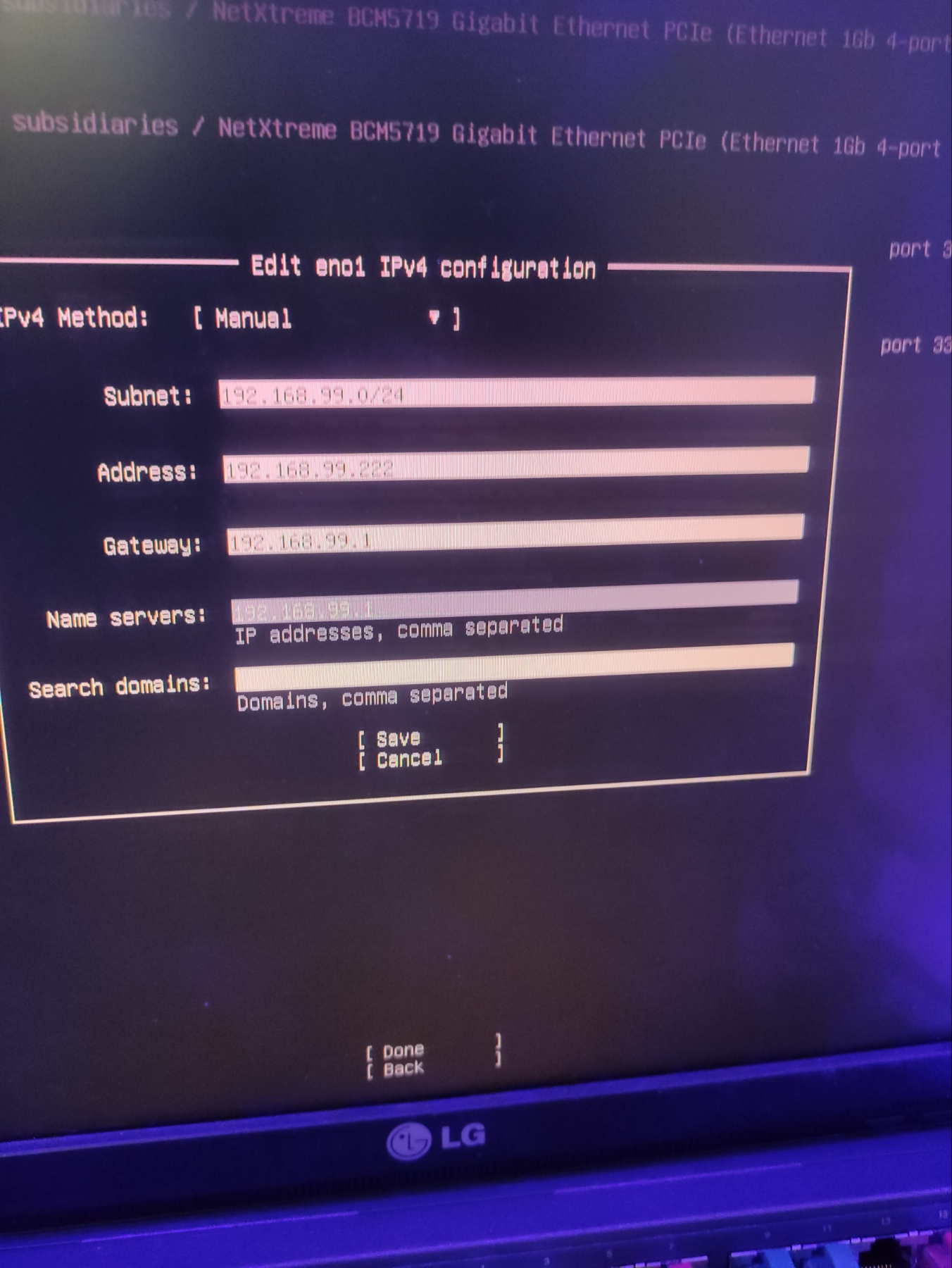
Избираме не минимална, а цяла инсталация:



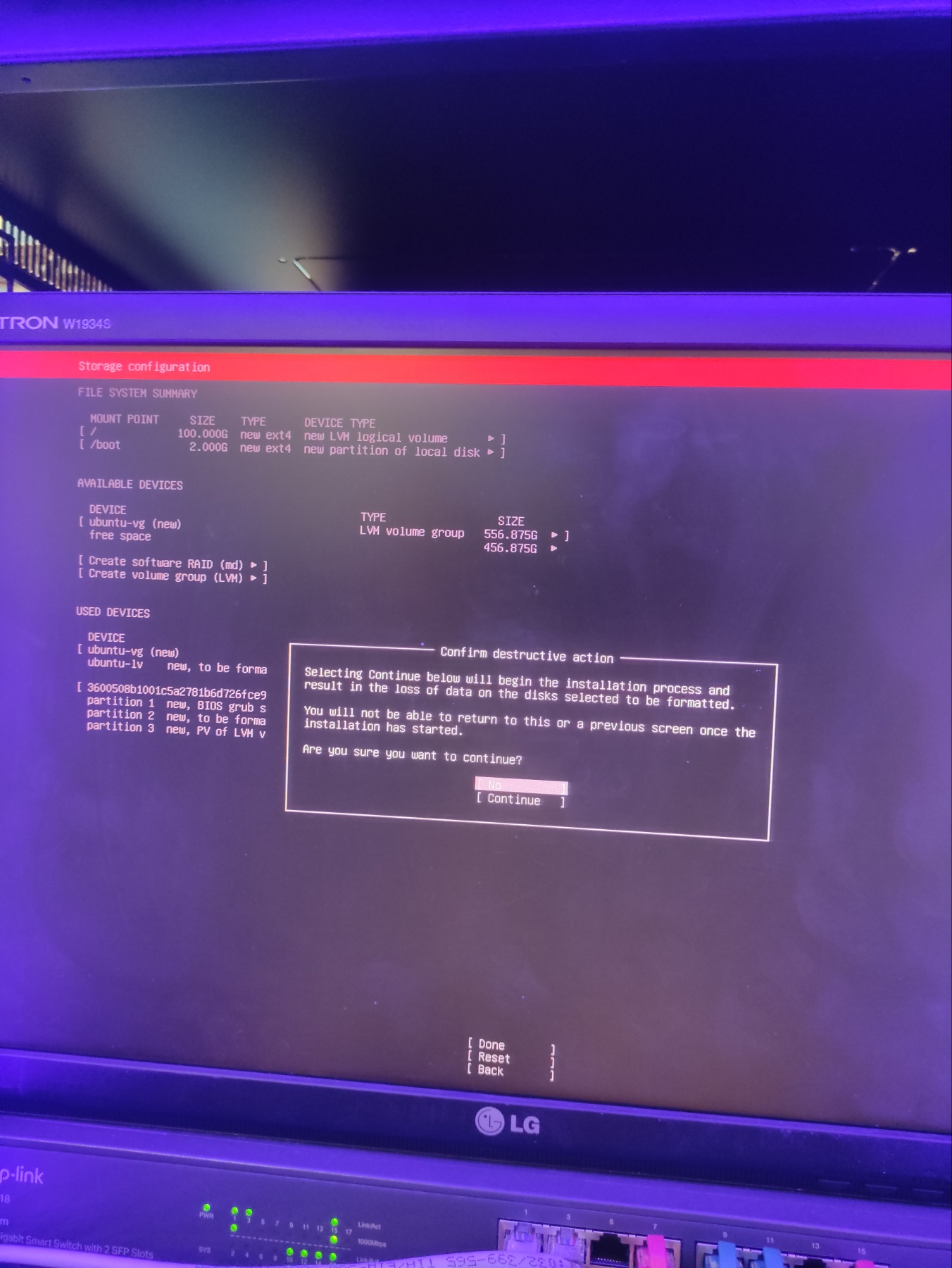
Идва ред на най важния момент да изберем статичен или динамичен IP адрес



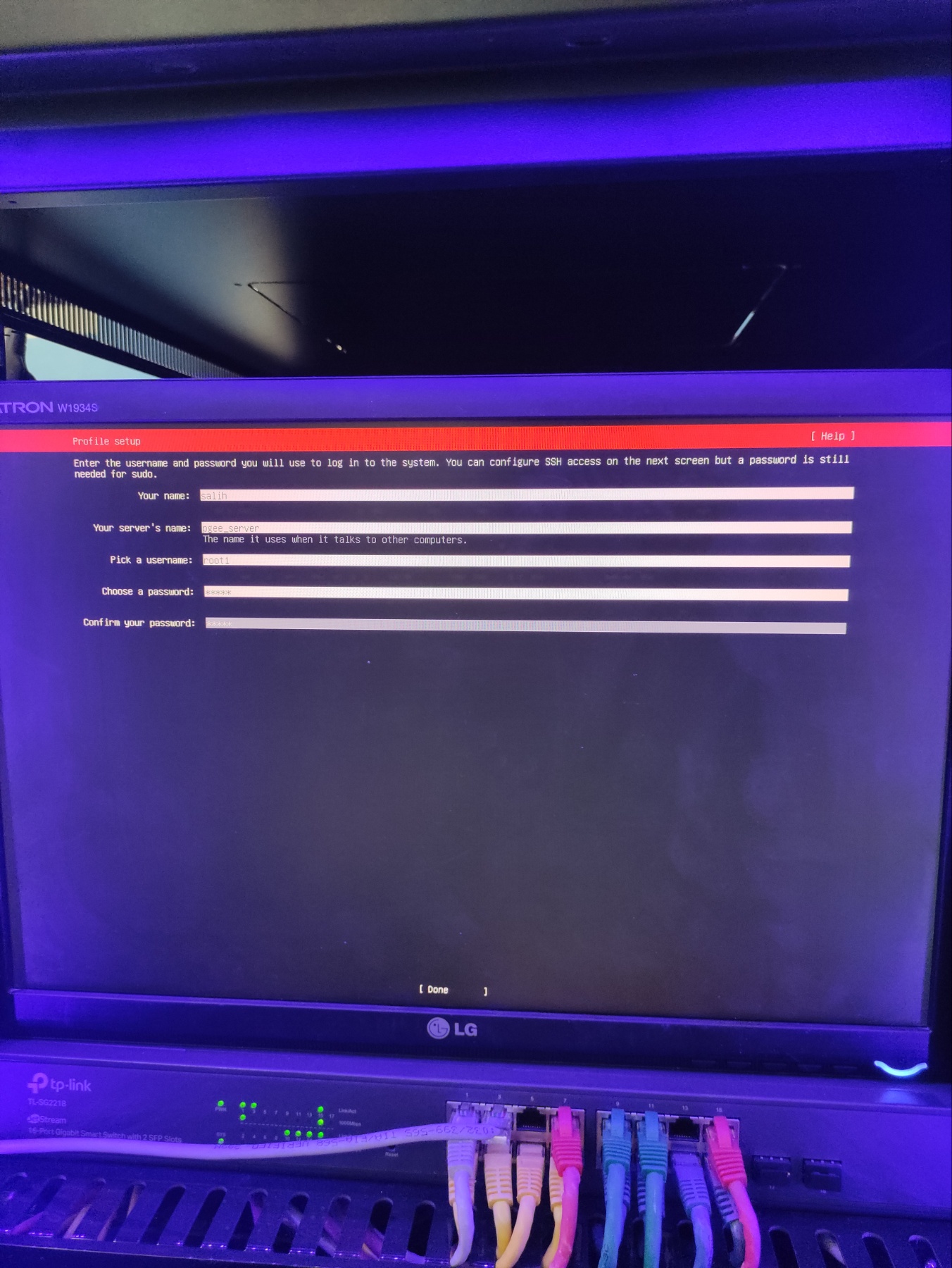


Избираме Edit IPv4 🡪 Manual 🡪 След това попълваме полетата с желаните адреси.

Още една важна стъпка е дали да делим дисковете. В нашия случай няма да го правим:



И на последно място, но не по важност е избирането на имена на сървър, потребител и парола която е важно да се запомни, защото после e много трудно се промени:



## Инсталиране на Python и Apache

Следващата стъпка е да проверим версията на python с командата:



Ако изпише, че имаме python2 трябва да инсталираме python3 с тази команда:



Но новите версии на Linux Ubuntu идват с python3.

Сега е време да инсталираме Apache със следната команда:



## Инсталиране на MySQL база данни

Трябва да вдигнем база данни, за да има къде да се запазва информация на сървъра. Най-популярното решение е MySQL. Да го инсталираме трябва да въведем следната команда:



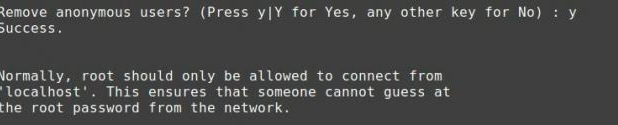
Натискаме „y“, за да продължим да инсталирането на MySQL сървъра.

Картина, която съдържа текст, Уебсайт

Описанието е генерирано автоматичноСлед като MySQL сървърът е успешно инсталиран, проверяваме инсталираната версия, като използваме командата:

Скриптът mysql\_secure\_installation, който се предлага по подразбиране с инсталацията MySQL, ни позволява да защитим MySQL. Стартираме скрипта mysql\_secure\_installation с командата:

По време на изпълнението на скрипта mysql\_secure\_installation ще се показват различни подкани. Първо ще видим три нива на политика за потвърждаване на паролата. Трябва да натиснем „y“, за да зададете парола. Освен това избираме нивото на паролата, като въведем съответната цифра, въвеждаме паролата. Командният ред ще покаже очакваната сила на паролата.

Ще бъдем подканени да премахнем анонимните потребители, да тестваме бази данни и да презаредим таблицата с привилегии. Натискаме 'y или n' според нашия избор и конфигурацията ще бъде извършена успешно.

За да започнем с MySQL сървъра от командния ред, използваме командата:

 Интерактивната обвивка MySQL ще стартира. След като MySQL сървърът е настроен, MySQL създава root потребител, който може да управлява базите данни и да извършва различни административни действия. Освен това може да се създаде нов потребител в MySQL сървъра. Докато създаваме нов потребител, въвеждаме потребителско име и парола. Следваме дадения по-долу синтаксис, за да създадем нов потребител:

След успешно създаване на новия потребител можем да предоставите привилегиите на потребителя, както следва:

Препоръчително е да изтрием привилегиите. Това ще освободи допълнителната памет, която сървърът кешира, докато създава нов потребител.

Картина, която съдържа текст

Описанието е генерирано автоматично

Сега нека влезем в MySQL сървъра и да създадем нова база данни. За да извършиме вход, използвайте командата:



За да създадем нова база данни, изпълняваме командата:



За да видим списъка с всички налични бази данни, въвеждаме командата:

## Инструкция за Python/Django

Като имаме вече и база данни няма какво да правим повече по сървъра, време е да си направим инструкцията за Python Django. Която е последната стъпка от нашата работа. Тази инструкция е качена на статичния IP адрес, който е 192.168.99.222. Ще я намерим, като въведем адреса в браузъра като google chrome. Там ще видите следното:

Инсталиране на пакети от хранилищата на Ubuntu

За да започнем процеса, ще изтеглим и инсталираме всички елементи, от които се нуждаем, от хранилищата на Ubuntu. Това ще включва уеб сървъра на Apach e, модула, използван за интерфейс с нашето приложение Django, и пакетния мениджър на Python, който може да се използва за изтегляне на нашите инструменти, свързани с Python.mod\_wsgipip

За да получите всичко, от което се нуждаем, актуализирайте локалния индекс на пакетите на вашия сървър и след това инсталирайте подходящите пакети.

Ако използвате Django с Python 2, командите, от които се нуждаете, са:

Картина, която съдържа текст

Описанието е генерирано автоматично

Ако вместо това използвате Django с Python 3, ще ви е необходим алтернативен модул и пакет Apache. Подходящите команди в този случай са: pip

Картина, която съдържа текст

Описанието е генерирано автоматично

Конфигуриране на виртуална среда на Python

Сега, когато имаме компонентите от хранилищата на Ubuntu, можем да започнем работа по нашия проект Django. Първата стъпка е да създадем виртуална среда на Python, така че нашият проект Django да бъде отделен от инструментите на системата и всички други проекти на Python, по които може да работим.

Трябва да инсталираме командата, за да създадем тези среди. Можем да използваме това .virtualenvpip

Ако използвате Python 2, въведете:



Ако използвате Python 3, въведете:



С инсталирането можем да започнем да формираме нашия проект. Създайте директория, където искате да запазите проекта си и да се преместите в директорията: virtualenv

Картина, която съдържа текст

Описанието е генерирано автоматично

В директорията на проекта създайте виртуална среда на Python, като въведете:



Това ще създаде директория, извикана във вашата директория. Вътре ще инсталира локална версия на Python и локална версия на . Можем да използваме това, за да инсталираме и конфигурираме изолирана Python среда за нашия проект .myprojectenvmyprojectpip

Преди да инсталираме изискванията на Python на нашия проект, трябва да активираме виртуалната среда. Можете да направите това, като въведете:



Вашата подкана трябва да се промени, за да покаже, че сега работите във виртуална среда на Python. Ще изглежда горе-долу така: .(myprojectenv)user@host:~/myproject$

С вашата виртуална среда активна, инсталирайте Django с локалния екземпляр на: pip

Независимо дали използвате Python 2 или Python 3, когато виртуалната среда е активирана, трябва да използваме командата (not ).pippip3



Създаване и конфигуриране на нов проект на Django

Сега, когато Django е инсталиран в нашата виртуална среда, можем да създадем действителните файлове на проекта Django.

Създаване на проекта Django

Тъй като вече имаме директория на проекта, ще кажем на Django да инсталира файловете тук. Той ще създаде директория от второ ниво с действителния код, който е нормален, и ще постави скрипт за управление в тази директория. Ключът към това е точката в края, която казва на Django да създаде файловете в текущата директория:



Регулиране на настройките на проекта

Първото нещо, което трябва да направим с нашите новосъздадени файлове на проекта, е да коригираме настройките. Отворете файла с настройките с вашия текстов редактор:



Ще използваме базата данни по подразбиране SQLite в това ръководство за по-голяма простота, така че всъщност не е нужно да променяме твърде много. Ще се съсредоточим върху конфигурирането на разрешените хостове да ограничават домейните, на които отговаряме, и конфигурирането на директорията със статични файлове, където Django ще постави статични файлове, така че уеб сървърът да може да ги обслужва лесно.

Започнете с намирането на линията. Вътре в квадратните скоби въведете публичния IP адрес на сървъра, името на домейна или и двете. Всяка стойност трябва да бъде обвита в кавички и разделена със запетая като нормален списък на Python:ALLOWED\_HOSTS

Картина, която съдържа текст

Описанието е генерирано автоматично

В долната част на файла ще добавим ред за конфигуриране на тази директория. Django използва настройката, за да определи директорията, където тези файлове трябва да отидат. Ще използваме малко Python, за да му кажем да използва директория, наречена "static" в главната директория на нашия проект: STATIC\_ROOT

Картина, която съдържа текст, Уебсайт

Описанието е генерирано автоматично

Запишете и затворете файла, когато приключите.

Завършване на първоначалната настройка на проекта

Сега можем да мигрираме първоначалната схема на базата данни към нашата база данни SQLite, като използваме скрипта за управление:

Картина, която съдържа текст

Описанието е генерирано автоматично

Създайте административен потребител за проекта, като въведете:



Ще трябва да изберете потребителско име, да предоставите имейл адрес и да изберете и потвърдите парола.

Можем да съберем цялото статично съдържание в местоположението на директорията, което конфигурирахме, като въведем:



Ще трябва да потвърдите операцията. Статичните файлове ще бъдат поставени в директория, извикана в директорията на вашия проект .static

Сега можем да регулираме настройките на защитната стена, за да позволим трафик към нашия сървър за разработка на Django, който ще работи на порт 8000. Ако сте следвали първоначалното ръководство за настройка на сървъра в предварителните условия, трябва да имате активирана защитна стена на UFW в момента.

Разрешаване на връзки към сървъра за разработка, като въведете:



И накрая, можете да тествате проекта си, като стартирате сървъра за разработка на Django с тази команда:

Картина, която съдържа текст

Описанието е генерирано автоматично

Във вашия уеб браузър посетете името на домейна или IP адреса на вашия сървър, последвано от::8000

Трябва да видите страницата с индекс по подразбиране на Django:



Ако добавите към края на URL адреса в адресната лента, ще бъдете подканени за административното потребителско име и парола, които сте създали с командата:/admincreatesuperuser

Картина, която съдържа текст

Описанието е генерирано автоматично

След удостоверяване можете да получите достъп до администраторския интерфейс по подразбиране на Django:



Когато приключите с проучването, натиснете CTRL-C в терминалния прозорец, за да изключите сървъра за разработка.

Вече сме приключили с Django за момента, така че можем да се отдръпнем от нашата виртуална среда, като напишем:



Конфигуриране на Apache

Сега, когато вашият проект Django работи, можем да конфигурираме Apache като преден край. Клиентските връзки, които получава, ще бъдат преведени във формат WSGI, който приложението Django очаква с помощта на модула. Това трябваше да бъде активирано автоматично при инсталирането по-рано .mod\_wsgi

За да конфигурираме WSGI паса, ще трябва да редактираме виртуалния хост файл по подразбиране:



Можем да запазим директивите, които вече присъстват в досието. Просто трябва да добавим някои допълнителни елементи.

За да започнете, нека конфигурираме статичните файлове. Ще използваме псевдоним, за да кажем на Apache да картографира всички заявки, започващи със "статичната" директория в папката на нашия проект. Събрахме статичните активи там по-рано. Ще настроим псевдонима и след това ще предоставим достъп до въпросната директория с блок от директории:/static

Картина, която съдържа текст

Описанието е генерирано автоматично

След това ще предоставим достъп до файла в директорията на проекта от второ ниво, където се съхранява кодът на Django. За да направите това, ще използваме секция с директория с файлова секция вътре. Ще предоставим достъп до файла вътре в тази вложена конструкция :wsgi.py

Картина, която съдържа текст

Описанието е генерирано автоматично

След като това е конфигурирано, ние сме готови да изградим частта от файла, която всъщност обработва WSGI паса. Ще използваме режим на демон, за да стартираме процеса WSGI, който е препоръчителната конфигурация. Можем да използваме директивата, за да установим това. WSGIDaemonProcess

Тази директива носи произволно име за процеса. Ще използваме, за да останем последователни. След това създадохме дома на Python, където Apache може да намери всички компоненти, които може да са необходими. Тъй като използвахме виртуална среда, можем да насочим това директно към нашата базова виртуална директория за среда. След това зададохме пътя на Python да сочи към основата на нашия проект Django.myproject

След това трябва да уточним групата на процеса. Това трябва да сочи към същото име, което избрахме за директивата (в нашия случай). И накрая, трябва да зададем псевдонима на скрипта, така че Apache да предава заявки за коренния домейн към файла: WSGIDaemonProcessmyprojectwsgi.py

Картина, която съдържа Уебсайт

Описанието е генерирано автоматично

Когато приключите с извършването на тези промени, запишете и затворете файла.

Приключване на някои проблеми с разрешенията

Ако използвате базата данни SQLite, която е използваната по подразбиране в тази статия, трябва да разрешите на процеса Apache достъп до този файл.

За да направите това, първата стъпка е да промените разрешенията, така че собственикът на групата на базата данни да може да чете и пише. Файлът на базата данни се извиква по подразбиране и трябва да се намира в директорията на базовия проект:db.sqlite3



След това трябва да дадем на групата Apache, групата, груповата собственост на файла :www-data



За да пишем във файла, трябва да дадем на групата Apache собственост върху родителската директория на базата данни:



Трябва отново да се приспособим през защитната стена. Вече не се нуждаем от отворен порт 8000, тъй като ние сме прокси чрез Apache, така че можем да премахнем това правило. След това можем да добавим изключение, за да позволим трафик към процеса на Apache:



Проверете вашите Apache файлове, за да се уверите, че не сте направили никакви синтактични грешки:



Докато последният ред на изхода изглежда така, вашите файлове са в добра форма:

Картина, която съдържа текст

Описанието е генерирано автоматично

След като тези стъпки са направени, вие сте готови да рестартирате услугата Apache, за да реализирате промените, които сте направили. Рестартирайте Apache, като въведете:



Сега трябва да имате достъп до вашия Django сайт, като отидете на името на домейна или IP адреса на вашия сървър, без да посочвате порт. Редовният сайт и администраторският интерфейс трябва да функционират според очакванията.

Следващи стъпки

След като проверите дали приложението ви е достъпно, е важно да защитите трафика към приложението си.

Ако имате име на домейн за вашето приложение, най-лесният начин да защитите приложението си е с безплатен SSL сертификат от Let's Encrypt. Следвайте нашето ръководство Let's Encrypt за Apache на 16.04, за да научите как да настроите това.

Ако нямате име на домейн за вашето приложение и го използвате за собствени цели или за тестване, винаги можете да създадете само подписан сертификат. Можете да научите как да настроите това с нашето ръководство за създаване на само подписани SSL сертификати за Apache на Ubuntu 16.04.

В това ръководство сме създали проект Django в собствена виртуална среда. Конфигурирахме Apache да обработва клиентски заявки и интерфейс с приложението Django .mod\_wsgi

Django прави създаването на проекти и приложения лесно, като предоставя много от общите части, което ви позволява да се съсредоточите върху уникалните елементи. Използвайки общата верига от инструменти, описана в тази статия, можете лесно да обслужвате приложенията, които създавате от един сървър.

# Заключение

Имам няколко съученика които имат за задача да правят Django проекти и те искат да имат достъп до проектите си извън училище. Затова моята задача е да направя училищен(учебен) уеб сървър. За целта ще ни е нужен подходящият софтуер и хардуер. Хардуера е предоставен от училището а за софтуера съм се спрял на Linux Ubuntu. Linux Ubuntu е една от най-популярните операционни системи за уеб сървъри, защото предлага няколко предимства за администриране на уеб сървъри-

* Отворен код
* Стабилност и надеждност
* Сигурност
* Лесен за използване

Терминът уеб сървър може да се отнася до хардуер или софтуер, или и двете От страна на хардуера, уеб сървърът е компютър, който съхранява софтуера на уеб сървъра и компонентните файлове на уебсайта (например HTML документи, изображения, CSS таблици със стилове и JavaScript файлове).

От страна на софтуера уеб сървърът включва няколко части, които контролират как уеб потребителите имат достъп до хоствани файлове. Като минимум това е HTTP сървър . HTTP сървърът е софтуер, който разбира URL адреси (уеб адреси) и HTTP (протоколът, който вашият браузър използва за преглед на уеб страници). HTTP сървърът може да бъде достъпен чрез имената на домейни на уебсайтовете, които съхранява, и той доставя съдържанието на тези хоствани уебсайтове до устройството на крайния потребител. За тези домейни си има цяла система. Тя се нарича DNS система. За цялата комуникация се грижи HTTP сървъра. Който разбира от HTTP заявки. Заявката я изпраща браузърът. Към сървърът, а сървърът връща отговори като **отговор**.

След като сме разгледали какво е уеб сървър и сме разбрали как той функционира следва да го инсталирам. След като сме избрали операционна система и нейната най-подходяща за нас, а именно Linux Ubuntu. Следва да я инсталирам на нашия предоставен от училището сървър. Или по-просто да инсталирам избрания софтуер на нашия хардуер.

Досега направихме вече сървър ако искаме да направим уеб сървър трябва да свалим Apache

Защо избрах Apache:

* + - Гъвкав, поради способността потребителите да избират различни модули.
    - Предлага по-голяма сигурност, което е очаквано, поради факта, че Windows – базираните машини са най-често подложени на атаки.
    - Ползва се с широка подкрепа от потребителската общност.
    - Работи под UNIX, Windows, Linux, Mac OS

Тези уебсайтове трябва да се съхраняват някъде. Трябва да изградим база данни най подходящата и най популярната в света е My SQL база данни.

Имам няколко съученика които имат за задача да правят Django проекти и те искат да имат достъп до проектите си извън училище. Затова моята задача е да направя училищен(учебен) уеб сървър. За целта ще ни е нужен подходящият софтуер и хардуер. Хардуера е предоставен от училището а за софтуера съм се спрял на Linux Ubuntu. Linux Ubuntu е една от най-популярните операционни системи за уеб сървъри, защото предлага няколко предимства за администриране на уеб сървъри-

* Отворен код
* Стабилност и надеждност
* Сигурност
* Лесен за използване

Терминът уеб сървър може да се отнася до хардуер или софтуер, или и двете От страна на хардуера, уеб сървърът е компютър, който съхранява софтуера на уеб сървъра и компонентните файлове на уебсайта (например HTML документи, изображения, CSS таблици със стилове и JavaScript файлове).

От страна на софтуера уеб сървърът включва няколко части, които контролират как уеб потребителите имат достъп до хоствани файлове. Като минимум това е HTTP сървър . HTTP сървърът е софтуер, който разбира URL адреси (уеб адреси) и HTTP (протоколът, който вашият браузър използва за преглед на уеб страници). HTTP сървърът може да бъде достъпен чрез имената на домейни на уебсайтовете, които съхранява, и той доставя съдържанието на тези хоствани уебсайтове до устройството на крайния потребител. За тези домейни си има цяла система. Тя се нарича DNS система. За цялата комуникация се грижи HTTP сървъра. Който разбира от HTTP заявки. Заявката я изпраща браузърът. Към сървърът, а сървърът връща отговори като **отговор**.

След като сме разгледали какво е уеб сървър и сме разбрали как той функционира следва да го инсталирам. След като сме избрали операционна система и нейната най-подходяща за нас, а именно Linux Ubuntu. Следва да я инсталирам на нашия предоставен от училището сървър. Или по-просто да инсталирам избрания софтуер на нашия хардуер.

Досега направихме вече сървър ако искаме да направим уеб сървър трябва да свалим Apache

Защо избрах Apache:

* + - Гъвкав, поради способността потребителите да избират различни модули.
    - Предлага по-голяма сигурност, което е очаквано, поради факта, че Windows – базираните машини са най-често подложени на атаки.
    - Ползва се с широка подкрепа от потребителската общност.
    - Работи под UNIX, Windows, Linux, Mac OS

Тези уебсайтове трябва да се съхраняват някъде. Трябва да изградим база данни най подходящата и най популярната в света е My SQL база данни.

# Източници и използвана литература

1. <https://help.superhosting.bg/what-is-web-server.html> какво е сървър , html и http
2. <https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_web_server> какво в сървър
3. <https://bg.theastrologypage.com/uniform-resource-locator> какво е URL?
4. <https://delta.bg/blog/predimstvata-da-imash-sobstven-ip-adres/> какво е Ip адрес?
5. <https://blog.superhosting.bg/dns-how-it-works-part1.html> какво е DNS?
6. <https://www.icn.bg/bg/blog/friendly-geeks-saveti/apache-vs-iis-da-izberem-pravilniyat-sarvar/> Предимства и недостатъци на Апач и М
7. <https://www.cloudns.net/blog/dhcp-server/> Сървър DHCP
8. <https://bg.strephonsays.com/what-is-the-difference-between-static-and-dynamic-ip-address> Динамичен и Статичен Ip адрес
9. <https://ciksiti.com/bg/chapters/2930-how-to-install-mysql-on-linux-mint-20-and-ubuntu-20-04---l> Кaк да инсталираме MySQL