Съдържание

[1. Увод 2](#_Toc195609976)

[2. Цел и задачите на дипломния проект 2](#_Toc195609977)

[2.1 Цел на проекта 2](#_Toc195609978)

[2.2 Задачи на проекта 3](#_Toc195609979)

[3. Теоритичен преглед 4](#_Toc195609980)

[3.1 Въведение в уеб разработката 4](#_Toc195609981)

[3.2 Сървърна архитектура и уеб приложения 4](#_Toc195609982)

[3.3 WSGI сървъри 5](#_Toc195609983)

[3.3.1 WSGI (Web Server Gateway Interface) 5](#_Toc195609984)

[3.3.2 Gunicorn 5](#_Toc195609985)

[3.3.3 uWSGI 5](#_Toc195609986)

[3.4 Уеб сървъри 6](#_Toc195609987)

[3.4.1 Nginx 6](#_Toc195609988)

[3.5 Django Framework 7](#_Toc195609989)

[3.6 Бази данни 7](#_Toc195609990)

[3.6.1 Инсталиране и конфигуриране на PostgreSQL 7](#_Toc195609991)

[4. Методология 8](#_Toc195609992)

[4.1 Планиране на проекта 8](#_Toc195609993)

[4.1.1 Определяне на целите 8](#_Toc195609994)

[4.1.2 Създаване на график 8](#_Toc195609995)

[4.1.3 Идентифициране на ресурси 9](#_Toc195609996)

[4.2 Подготовка на сървърната среда 9](#_Toc195609997)

[4.2.1 Избор на хостинг 9](#_Toc195609998)

[4.2.2 Инсталиране на операционната система 9](#_Toc195609999)

[4.2.3 Инсталиране на основни пакети 9](#_Toc195610000)

[5. Практическа част 10](#_Toc195610001)

[5.1 Инсталация на необходимия софтуер 10](#_Toc195610002)

[5.1.1 Инсталиране на Ubuntu Server 10](#_Toc195610003)

[5.1.2 Инсталиране на необходими пакети 10](#_Toc195610004)

[5.2 Настройка на MySQL 11](#_Toc195610005)

[5.2.1 Създаване на MySQL база данни и потребител 11](#_Toc195610006)

[5.3 Инсталиране и конфигуриране на Django 12](#_Toc195610007)

[5.4 Настройка на Nginx 14](#_Toc195610008)

[5.4.1 Конфигуриране на Nginx 14](#_Toc195610009)

[6. Заключение 14](#_Toc195610010)

[7. Литература: 15](#_Toc195610011)

# Увод

# Цел и задачите на дипломния проект

## Цел на проекта

Целта на настоящия дипломен проект е да се разработи, конфигурира и деплойне уеб приложение, използвайки технологията Django и средата на операционната система Ubuntu Server. Проектът има за цел не само техническата реализация на уеб приложението, но и разширяване и задълбочаване на знанията в областта на уеб разработката, управление на сървъри и работа с различни инструменти за конфигурация на уеб среда.

По-конкретно, проектът предвижда изграждането на работещо приложение, което ще демонстрира основните принципи на структуриране на уеб приложения, работа с бази данни, обработка на HTTP заявки и взаимодействие с уеб сървъри. Също така, той ще осигури основа за разбиране на различията между локална и реална среда, което е критична част от професионализацията в областта на софтуерното инженерство и уеб разработката.

## Задачи на проекта

Да се реализира поставената цел, проектът включва следните основни задачи:

**Анализ на съществуващите решения**:

* + Изследване на различни методи и подходи за хостинг на уеб приложения, с фокус върху популярната технология за работа с уеб сървъри и WSGI сървъри, като Gunicorn и uWSGI.
  + Комплексен анализ на работата на Nginx, който е един от най-разпространените уеб сървъри, използвани за обработка на HTTP заявки и като реверсивен прокси.

1. **Подготовка на сървърната среда**:
   * Провеждане на инсталация на операционна система Ubuntu Server, включително избора на необходимите конфигурации и изисквания за работа на приложението.
   * Изготвяне на подробен план за инсталиране и конфигуриране на Nginx, който позволява на приложението да бъде достъпно за потребители и да обработва заявките правилно.
2. **Разработка и деплой на Django приложение**:
   * Изграждане на прост Django проект, който да служи като демонстрация на функционалността на платформата и проверки на различни компоненти.
   * Инсталиране на необходимите зависимости, конфигуриране на базата данни и създаване на модели, които да отразяват данните, с които приложението ще работи.
3. **Тестване на конфигурацията**:
   * Провеждане на тестове, за да се уверим, че приложението функционира безпроблемно в реалната сървърна среда. Това ще включва генериране на HTTP заявки и наблюдение на отговорите, анализ на производителността и времето за отговор.
4. **Документиране на процеса**:
   * Подготвяне на изчерпателна документация, която да описва стъпките на инсталация и конфигуриране на приложението, както и проблемите, с които е било срещнато и разрешаване на тях. Тази документация ще служи не само за учебна цел, а и за ръководство за бъдещи потребители и разработчици на приложението.

# Теоритичен преглед

## Въведение в уеб разработката

Уеб разработката е сложна и многофункционална област, която обхваща множество аспекти от дизайна до управлението на сървъри. Днешните уеб приложения съчетават повече от просто статични страници; те съдържат динамични елементи, интерактивност и свързани бекенд системи. Уеб разработката се разделя на две основни категории: **frontend** (клиентска част) и **backend** (сървърна част).

* **Frontend разработка**: Включва всичко, което потребителите виждат и взаимодействат с, като HTML, CSS и JavaScript. Фронтенд разработчиците се концентрират върху потребителския опит и визуалната презентация на уеб приложенията.
* **Backend разработка**: Състои се от сървърната част на уеб приложението, която обработва логиката, базите данни и бизнес правилата. Тук влизат езици за програмиране като Python, PHP, Ruby, Java и много други.

## Сървърна архитектура и уеб приложения

Уеб приложенията функционират чрез модулна архитектура, включваща три основни слоя:

1. **Клиентска част (Frontend)**: Тази част включва потребителски интерфейс, реализиран чрез HTML, CSS и JavaScript. Разработчиците използват библиотеки и фреймворкове като React, Angular и Vue.js, за да създадат интерактивни приложения.
2. **Сървърна част (Backend)**: Тук се осъществява логиката на приложението, обработката на данни и взаимодействието с бази данни. Често се използват фреймворкове като Django (за Python), Laravel (за PHP) иRuby on Rails (за Ruby) за бързо и ефективно разработване на приложения.
3. **Бази данни**: Уеб приложенията съхраняват и манипулират данни, основно чрез релационни бази данни (както PostgreSQL и MySQL) и NoSQL решения (като MongoDB). Всеки от тези типове бази данни предлага различни предимства в зависимост от нуждите на приложението.

## WSGI сървъри

### WSGI (Web Server Gateway Interface)

WSGI е стандарт за комуникация между Python приложения и уеб сървъри. Той е основополагающият интерфейс, който позволява на Python приложения да функционират на различни уеб платформи. WSGI предлага единен начин за интегриране на Python приложения с уеб сървъри, което осигурява гъвкавост и избираемост на технологиите.

### Gunicorn

**Gunicorn** (Green Unicorn) е WSGI HTTP сървър, предназначен за UNIX-системи. Той поддържа множество работници (workers), което позволява паралелна обработка на многобройни HTTP заявки. Gunicorn е идеален за приложения, разработени с Django, и предлага висока производителност в сравнение с много от съществуващите сървъри. Настройването на Gunicorn е просто и започва с напътствия, които включват:

bash

gunicorn myproject.wsgi:application

Тази команда стартира Gunicorn и указва на него WSGI приложението, което иска да обслужва.

### uWSGI

**uWSGI** е друг популярен WSGI сървър, който предлага изключителна гъвкавост и функции. Той е важен компонент в настройката на Python приложения и поддържа различни протоколи за свързване с уеб сървъри. uWSGI има много настройки и опции, което може да го направи по-сложен за конфигуриране в сравнение с Gunicorn.

## Уеб сървъри

### Nginx

**Nginx** е един от най-разпространените и мощни уеб сървъри в света. Той е проектиран за бързина, надеждност и ефективност при обработката на множество паралелни заявки. Nginx е известен със своята способност да функционира като реверсивен прокси и също така да бъде load balancer. Успешната конфигурация на Nginx е от изключителна важност за правилното функциониране на Django приложения, а основните стъпки включват:

1. **Инсталация на Nginx**:

bash

sudo apt-get install nginx

1. **Базова конфигурация**:  
   Конфигурационният файл на Nginx обикновено се намира в /etc/nginx/sites-available/. Основните настройки включват указването на порта, настройките за вашия домейн и директорията за статични файлове. Примерна конфигурация изглежда така:

nginx

server {

listen 80;

server\_name example.com;

location /static/ {

alias /path/to/staticfiles/;

}

location / {

proxy\_pass http://localhost:8000;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

proxy\_set\_header Host $host;

proxy\_cache\_bypass $http\_upgrade;

}

}

1. **Активиране и рестартиране на Nginx**:  
   За да активирате новата конфигурация, се създава символна връзка и Nginx се рестартира:

bash

sudo ln -s /etc/nginx/sites-available/example /etc/nginx/sites-enabled/

sudo systemctl restart nginx

## Django Framework

Django е мощен фреймворк с отворен код, написан на Python, предназначен за бързо разработване на уеб приложения. Неговата архитектура следва патерна Model-View-Template (MVT), който позволява чиста организация на кода. Основни характеристики на Django включват:

* **Административен панел**: Автоматично генерираният административен интерфейс дава възможност за удобно управление на съдържанието.
* **ORM (Object-Relational Mapping)**: Позволява взаимодействие с бази данни чрез Python обекти, което прави работата с бази данни по-разбираема и удобна.
* **Сигурност**: Django вгражда множество механизми за защитаване на приложения от общи заплахи за сигурност, като CSRF, SQL инжекции и XSS.

## Бази данни

В проектите с Django се използват различни бази данни, но **PostgreSQL** е на особена почит поради своите функции, производителност и разширяемост. PostgreSQL е релационна база данни, известна със своята солидност и разнообразие от функции.

### Инсталиране и конфигуриране на PostgreSQL

1. **Инсталация**:

bash

sudo apt-get install postgresql postgresql-contrib

1. **Създаване на база данни**:
   * Влизане в PostgreSQL:

bash

sudo -u postgres psql

* + Създаване на нова база данни:

sql

CREATE DATABASE mydatabase;

1. **Свързване на Django с PostgreSQL**:  
   В settings.py на Django, конфигурацията за базата данни трябва да бъде настроена, както следва:

python

DATABASES = {

'default': {

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

'NAME': 'mydatabase',

'USER': 'myuser',

'PASSWORD': 'mypassword',

'HOST': 'localhost',

'PORT': '',

}

}

# Методология

Методологията на дипломния проект се състои от систематичен и структурен подход към разработването на уеб приложение, което да бъде внедрено на реален сървър с операционна система Ubuntu. Проектът е организиран в логическа последователност, включваща кратък анализ на целите, изискванията, подготовката на средата, разработката на приложението и финализацията чрез тестване. Всеки етап е критичен за постигане на успешен резултат.

## Планиране на проекта

### Определяне на целите

Основната цел на дипломния проект е да се демонстрира умението за качване и конфигуриране на Django приложение на реален уеб сървър. Важно е целите да бъдат конкретни и количествени, например:

* Да създадем функционално Django приложение.
* Да конфигурираме Nginx за обслужване на приложението.
* Да се научим как да използваме WSGI сървър за интеграция на приложението с уеб сървъра.

### Създаване на график

Графикът е важен инструмент за управление на времето, който включва всички етапи на проекта. Всеки етап трябва да има определен срок. Например:

* Планиране: 1 седмица.
* Подготовка на сървърната среда: 2 седмици.
* Разработка на приложението: 3 седмици.
* Тестове и финална настройка: 1 седмица.

Този график не само че помага в организирането на времето, но и осигурява отчетност за напредъка на проекта.

### Идентифициране на ресурси

Необходимо е да се идентифицират ресурсите, нужни за успешна реализация на проекта. Тези ресурси могат да включват:

* Технически средства: компютри, сървъри, софтуерни инструменти.
* Човешки ресурси: участващи лица, консултанти, програматори или приятели, които да предоставят съвети.
* Финансови ресурси: средства за хостинг услуги или софтуерни лицензи, ако е необходимо.

## Подготовка на сървърната среда

След планирането, екипът преминава към подготовката на сървърната среда. Този процес може да включва следните основни стъпки:

### Избор на хостинг

Изборът на подходящ хостинг е от изключителна важност, тъй като той трябва да осигури достатъчно ресурси и поддръжка. Важно е да се избегнат платформи, които не са съвместими с използваните технологии.

### Инсталиране на операционната система

Инсталирането на Ubuntu Server включва следните основни стъпки:

* Изтегляне на ISO файла на Ubuntu.
* Записване на файла на USB устройство с помощта на инструменти като Rufus.
* Стартиране от USB и следване на инструкциите за инсталиране на операционната система.
* Конфигуриране на основни параметри, като език и мрежови настройки.

### Инсталиране на основни пакети

След инсталацията на операционната система, е необходимо да се инсталират основни пакети, например:

* Nginx: За управление на HTTP заявките.
* Python и pip: За работа с Django.
* PostgreSQL: За управление на бази данни.

Тези пакети ще формират основата на сървърната среда

# Практическа част

Практическата част на дипломния проект е фокусирана върху конкретните стъпки за инсталиране на софтуер, конфигуриране на средата, разработка и тестване на уеб приложението. Систематичният подход осигурява успешното прилагане на всички необходими стъпки.

## Инсталация на необходимия софтуер

### Инсталиране на Ubuntu Server

Първата стъпка в практическата част е инсталирането на операционната система Ubuntu Server, която ще служи за основа на нашето уеб приложение.

1. **Изтегляне на ISO файл**:  
   Изтегляме ISO файл на Ubuntu Server от [официалния сайт на Ubuntu](https://ubuntu.com/download/server). Препоръчително е да изберем версия с дългосрочна поддръжка (LTS).
2. **Подготовка на USB инсталационен носител**:  
   Използваме софтуер като **Rufus** (за Windows) или **UNetbootin** (за Linux), за да запишем ISO файла на USB устройство.
3. **Стартиране на инсталацията**:
   * Настройваме BIOS/UEFI на компютъра, за да стартира от USB устройството.
   * Избираме предпочитания език за инсталация, мрежови настройки и инсталация на OpenSSH Server.

### Инсталиране на необходими пакети

След успешната инсталация на Ubuntu Server, инсталираме необходимите пакети, включително MySQL.

1. **Обновление на системата**:

bash

sudo apt update && sudo apt upgrade -y

1. **Инсталиране на Python и pip**:

bash

sudo apt install python3 python3-pip -y

1. **Инсталиране на MySQL**:  
   Ние инсталираме MySQL, каквато ще е релационната база данни:

bash

sudo apt install mysql-server -y

По време на инсталацията ще бъде необходимо да конфигурираме MySQL, включително задаване на root парола.

1. **Инсталиране на Nginx**:

bash

sudo apt install nginx -y

1. **Инсталиране на Gunicorn**:

bash

pip install gunicorn

## Настройка на MySQL

След инсталирането на необходимите софтуерни компоненти, следва конфигурация на MySQL.

### Създаване на MySQL база данни и потребител

1. **Стартиране на MySQL сървъра**:  
   Дори и инсталацията по подразбиране да стартира MySQL, е добре да го проверим:

bash

sudo systemctl start mysql

1. **Влизане в MySQL**:  
   За достъп до MySQL:

bash

sudo mysql -u root -p

Въведете паролата за root, когато бъдете подканени.

1. **Създаване на база данни**:  
   В MySQL конзолата можем да създадем база данни и потребител:

sql

CREATE DATABASE mydatabase;

CREATE USER 'myuser'@'localhost' IDENTIFIED BY 'mypassword';

GRANT ALL PRIVILEGES ON mydatabase.\* TO 'myuser'@'localhost';

FLUSH PRIVILEGES;

1. **Изход от MySQL**:

sql

EXIT;

## Инсталиране и конфигуриране на Django

След конфигурирането на MySQL, следва инсталирането на Django и неговата настройка за работа с базата данни.

1. **Инсталиране на Django и MySQL Connector**:  
   За да интегрираме Django с MySQL, първо инсталираме необходимите библиотеки:

bash

pip install django mysqlclient

Библиотеката mysqlclient е необходима, за да позволи на Django да работи с MySQL.

1. **Създаване на нов Django проект**:  
   Създаваме нов проект, наречен myproject, с командата:

bash

django-admin startproject myproject

Тази команда създава директория myproject, която включва основните файлове на проекта.

1. **Конфигурация на файл settings.py**:  
   След успешното създаване на проекта, отваряме файла settings.py, намиращ се в директорията myproject, и правим следните изменения:
   * Променяме конфигурацията на базата данни, за да изглежда така:

python

DATABASES = {

'default': {

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

'NAME': 'mydatabase',

'USER': 'myuser',

'PASSWORD': 'mypassword',

'HOST': 'localhost',

'PORT': '3306',

}

}

1. **Добавяне на инсталации**:  
   Уверяваме се, че приложението django.contrib.staticfiles е включено в списъка на инсталираните приложения в settings.py, тъй като ще работим със статични файлове:

python

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

# добавете своите приложения тук

]

1. **Миграция на базата данни**:  
   За да създадем необходимите таблици в MySQL, необходимо е да извършим миграция:

bash

python3 manage.py migrate

Тази команда ще създаде таблиците, нужни за работа на Django с базата данни.

1. **Стартиране на сървъра за тестване**:  
   Можем да стартираме вграден сървър на Django, за да тестваме приложението:

bash

python3 manage.py runserver 0.0.0.0:8000

Сега приложението трябва да е достъпно на IP адреса на вашия сървър на порт 8000. За да тествате, можете да отворите браузър и да въведете http://<SERVER\_IP>:8000.

1. **Създаване на суперпотребител**:  
   За да имаме достъп до административния панел на Django, трябва да създадем суперпотребител:

bash

python3 manage.py createsuperuser

Следвайки указанията, задайте потребителско име, имейл и парола.

## Настройка на Nginx

След успешното създаване и тестване на Django приложението, следва конфигурацията на Nginx, който ще служи като уеб сървър.

### Конфигуриране на Nginx

1. **Създаване на конфигурационен файл за Nginx**:  
   Създайте нов конфигурационен файл за сайта:

bash

sudo nano /etc/nginx/sites-available/myproject

# Заключение

# Литература:

1. Колисниченко, Денис, Адаптивен уеб дизайн с Bootstrap, Асеневци, С 2019
2. D.K Academy, Python – практическо програмиране. Асеневци, С 2022
3. Донълдсън, Тоби. Бързо ръководство:Програмиране с Python.АлексСофт, С 2017
4. Есканази, Аврам. Софтуерни техологии. КЛМН, С 2006
5. https://softuni.bg/blog/what-is-django
6. 2. https://cynoteck.com/bg/blog-post/flask-vs-django/
7. 3. https://bg.savtec.org/articles/coding/the-basics-of-rest-and-restful-api-development.html
8. 4. https://urocibg.eu/
9. 5. https://www.django-rest-framework.org/
10. 6. https://softuni.bg/blog/vue-js-january-2021
11. 7. https://bg.wikipedia.org/wiki/MySQL
12. 8. https://priobshti.se/article/strategii-v-pomosht-na-prepodavaneto/taksonomiya-na-blum-ili-kak-da-napravim-uroka-v
13. 9. https://html.w3schools.bg
14. 10. https://www.inventum.bg/web-design/css/