MyGest

Guida Completa: File Storage e Deploy

Architettura, Implementazione e Procedure

Versione: 1.0

Data: 17 Novembre 2025

Autore: Sistema MyGest

Guida Completa: File Storage e Deploy MyGest

Data: 17 Novembre 2025

PARTE 1: ARCHITETTURA FILE STORAGE

1.1 Overview del Sistema

MyGest utilizza un sistema di storage centralizzato basato su NAS (Network Attached Storage) per gestire tutti i file dell'applicazione. Il sistema è progettato per funzionare sia in ambiente locale (WSL con mount NAS) che in produzione (VPS con mount NFS).

Caratteristiche Principali

- **Storage Unico**: Tutti i file sono archiviati in un'unica posizione configurabile
- **Struttura Gerarchica**: Organizzazione per cliente → titolario → anno
- **Storage Dinamico**: Path configurabile tramite variabile d'ambiente
- **Pattern di Naming**: Sistema flessibile per rinominare file automaticamente
- **Gestione Titolario**: Supporto per classificazione documenti con voce di default

1.2 Struttura Directory Archivio

Schema Gerarchico

```
ARCHIVIO_BASE_PATH/
├── {cliente_slug}/
│   ├── {titolario_path}/
│   │   ├── {anno}/
│   │   │   ├── documento_001.pdf
│   │   │   ├── documento_002.pdf
│   │   │   └── ...
│   │   └── {altro_anno}/
│   └── {altro_titolario}/
├── {altro_cliente}/
└── archivio_fisico/
    └── operazioni/
        ├── verbale_123.pdf
        └── ...
```

Componenti del Path

1. **ARCHIVIO_BASE_PATH**: Radice dello storage
2. Locale: `/mnt/archivio`
3. Produzione: `/srv/mygest/archivio`
4. **cliente_slug**: Slug univoco del cliente
5. Esempio: `rossi-spa`, `bianchi-mario`
6. **titolario_path**: Path gerarchico della voce di titolario
7. Esempio: `01-Contabilità`, `02-Fiscale/02.01-IVA`
8. Default: `99-Varie` (per documenti senza titolario)
9. **anno**: Anno di competenza del documento
10. Esempio: `2025`, `2024`

Esempio Concreto

```
/mnt/archivio/
├── rossi-spa/
│   ├── 01-Contabilità/
│   │   ├── 2024/
│   │   │   ├── FAT_20241115_001.pdf
│   │   │   └── DDT_20241120_002.pdf
│   │   └── 2025/
│   │       └── FAT_20250115_003.pdf
│   ├── 02-Fiscale/
│   │   └── 02.01-IVA/
│   │       └── 2025/
│   │           └── DICHIARAZIONE_20250301.pdf
│   └── 99-Varie/
│       └── 2025/
│           └── TEMP_20250117_documento.pdf
└── bianchi-mario/
    └── 01-Contabilità/
        └── 2025/
            └── RIC_20250110_001.pdf
```

1.3 Configurazione Storage

File di Configurazione

`.env` (Locale - WSL)

```
# Database
DATABASE_URL=postgresql://mygest_user:password@localhost:5432/mygest

# Storage - Mount NAS locale
ARCHIVIO_BASE_PATH=/mnt/archivio

# Django
SECRET_KEY=your-local-secret-key
DEBUG=True
ALLOWED_HOSTS=localhost,127.0.0.1
```

`.env.production` (VPS)

```
# Database
DATABASE_URL=postgresql://mygest_user:secure_password@localhost:5432/mygest_prod

# Storage - Mount NFS su VPS
ARCHIVIO_BASE_PATH=/srv/mygest/archivio

# Django
SECRET_KEY=your-production-secret-key-very-long-and-secure
DEBUG=False
ALLOWED_HOSTS=yourdomain.com,www.yourdomain.com,your-vps-ip
```

settings.py

```
import os
from pathlib import Path

# Carica variabili d'ambiente
from dotenv import load_dotenv
load_dotenv()

# Storage Configuration
ARCHIVIO_BASE_PATH = os.getenv("ARCHIVIO_BASE_PATH", "/mnt/archivio")

# MEDIA_ROOT unificato con ARCHIVIO
MEDIA_ROOT = ARCHIVIO_BASE_PATH
MEDIA_URL = "/archivio/"

# Configurazione Django Storages
DEFAULT_FILE_STORAGE = "mygest.storages.NASPathStorage"
```

1.4 Classe NASPathStorage

Implementazione

File: `mygest/storages.py`

```
from django.core.files.storage import FileSystemStorage
from django.conf import settings


class NASPathStorage(FileSystemStorage):
    """
    Storage personalizzato che legge dinamicamente il path da settings.
    Permette di cambiare la root dello storage senza riavviare l'applicazione.
    """

    @property
    def location(self):
        """Legge il path dello storage da settings ad ogni accesso"""
        return settings.ARCHIVIO_BASE_PATH

    @property
    def base_url(self):
        """URL base per accedere ai file"""
        return settings.MEDIA_URL


# Istanza globale dello storage
nas_storage = NASPathStorage()
```

Caratteristiche

1. **Dinamicità**: Il path viene letto ogni volta da `settings.ARCHIVIO_BASE_PATH`
2. **Flessibilità**: Cambiando `.env` si cambia lo storage senza modificare codice
3. **Compatibilità**: Estende `FileSystemStorage` di Django
4. **Thread-Safe**: Usa property per lettura runtime del path

1.5 Modello Documento

Schema del Modello

File: `documenti/models.py`

```
from django.db import models, transaction
from mygest.storages import nas_storage
import os
import logging

logger = logging.getLogger(__name__)

User = get_user_model()


def get_or_create_default_titolario():
    """
    Ottiene (o crea) la voce di titolario di default '99 - Varie'
    per i documenti senza titolario assegnato.
    """
    from titolario.models import TitolarioVoce

    voce, created = TitolarioVoce.objects.get_or_create(
        codice="99",
        parent__isnull=True,
        defaults={
            "descrizione": "Varie",
            "note": "Voce di default per documenti senza titolario",
        }
    )
    if created:
        logger.info("Creata voce di titolario default: 99 - Varie")
    return voce


class Documento(models.Model):
    # ... altri campi ...

    cliente = models.ForeignKey(
        "anagrafiche.Cliente",
        on_delete=models.CASCADE,
        related_name="documenti",
    )

    titolario_voce = models.ForeignKey(
        "titolario.TitolarioVoce",
        on_delete=models.SET_NULL,
        null=True,
        blank=True,
        related_name="documenti",
    )

    file = models.FileField(
        storage=nas_storage,
        upload_to="documenti/temp",
        blank=True,
        null=True,
    )

    percorso_archivio = models.CharField(
        max_length=512,
```

```
        blank=True,
        help_text="Path assoluto della directory su NAS"
    )

    anno_competenza = models.IntegerField(
        blank=True,
        null=True,
        help_text="Anno di competenza fiscale/contabile"
    )
```

1.6 Logica di Costruzione Path

Metodo _build_path()

```
def _build_path(self) -> str:
    """
    Costruisce il percorso assoluto della cartella di archivio.
    Se il documento non ha titolario, usa il default "99 - Varie".

    Ritorna path nel formato:
    /mnt/archivio/{cliente_slug}/{titolario_path}/{anno}/
    """
    from fascicoli.utils import build_titolario_parts

    # Usa titolario del documento o default "99 - Varie"
    voce_da_usare = self.titolario_voce
    if not voce_da_usare:
        voce_da_usare = get_or_create_default_titolario()

    anno = self.anno_competenza
    cliente = self.cliente

    if not anno or not cliente:
        return settings.ARCHIVIO_BASE_PATH

    # Costruisce path gerarchico del titolario
    # Es: "01-Contabilità" o "02-Fiscale/02.01-IVA"
    titolario_path = build_titolario_parts(voce_da_usare)

    parts = [
        settings.ARCHIVIO_BASE_PATH,  # /mnt/archivio
        cliente.slug,                  # rossi-spa
        titolario_path,                # 01-Contabilità
        str(anno),                     # 2025
    ]
    return os.path.join(*parts)
```

Funzione build_titolario_parts()

File: `fascicoli/utils.py`

```
def build_titolario_parts(voce: "TitolarioVoce") -> str:
    """
    Costruisce il path gerarchico di una voce di titolario.
    Percorre l'albero dal nodo verso la radice.

    Esempio:
        Input: voce "IVA" (codice 02.01, parent "Fiscale" codice 02)
        Output: "02-Fiscale/02.01-IVA"
    """
    parts = []
    current = voce

    while current:
        # Formato: "{codice}-{descrizione}"
        part = f"{current.codice}-{current.descrizione}"
        parts.insert(0, part)  # Inserisce all'inizio
        current = current.parent

    return os.path.join(*parts) if parts else ""
```

1.7 Sistema di Naming Pattern

Pattern Configurabili

File: `documenti/models.py`

```
PATTERN_FIELDS = {
    "tipo.codice": "Codice del tipo documento (es: FAT, DDT)",
    "data_documento": "Data documento (formato configurabile)",
    "codice": "Codice progressivo del documento",
    "cliente.ragione_sociale": "Ragione sociale del cliente",
    "attr:field_name": "Qualsiasi attributo del modello",
    "slug:field_name": "Versione slugificata di un campo",
}
```

Esempi di Pattern

```
# Pattern 1: {tipo.codice}_{data_documento:%Y%m%d}_{codice}
# Risultato: FAT_20250117_001.pdf

# Pattern 2: {data_documento:%Y%m%d}_{slug:titolo}
# Risultato: 20250117_fattura-cliente-rossi.pdf

# Pattern 3: {cliente.ragione_sociale}_{tipo.codice}_{codice}
# Risultato: RossiSpa_FAT_001.pdf
```

Implementazione

```
def build_document_filename(doc_instance, original_filename: str) -> str:
    """
    Costruisce il nome del file secondo il pattern configurato.
    """
    pattern = getattr(settings, "DOCUMENTI_FILENAME_PATTERN",
                      "{tipo.codice}_{data_documento:%Y%m%d}_{codice}")

    # Parsing del pattern
    # Sostituzione dei token
    # Gestione slug e attributi custom

    ext = os.path.splitext(original_filename)[1]
    return f"{result}{ext}"
```

1.8 Gestione Titolario e Spostamento File

Logica del Titolario

1. **Documenti senza titolario**: Vanno automaticamente in "99 - Varie"
2. **Documenti fascicolati**: Possono avere titolario diverso dal fascicolo
3. **Cambio titolario**: Il file viene spostato nella nuova directory

Metodo save() con Gestione Spostamento

```
@transaction.atomic
def save(self, *args, **kwargs):
    is_new = self.pk is None
    original_name = None
    if self.file and hasattr(self.file, "name"):
        original_name = os.path.basename(self.file.name)

    # Verifica se il titolario è cambiato (per spostare il file)
    titolario_changed = False
    old_percorso = None
    if not is_new and self.pk:
        try:
            old_doc = type(self).objects.only(
                "titolario_voce_id",
                "percorso_archivio"
            ).get(pk=self.pk)

            if old_doc.titolario_voce_id != self.titolario_voce_id:
                titolario_changed = True
                old_percorso = old_doc.percorso_archivio
                logger.info(
                    "Documento id=%s: titolario_voce cambiato da %s a %s",
                    self.pk,
                    old_doc.titolario_voce_id,
                    self.titolario_voce_id
                )
        except type(self).DoesNotExist:
            pass

    if is_new and not self.codice:
        seq = self._next_seq()
        self.codice = self._generate_codice(seq)

    # Ricalcola percorso_archivio (usa default se necessario)
    self.percorso_archivio = self._build_path()

    super().save(*args, **kwargs)

    # Rinomina file secondo pattern
    if self.file and original_name:
        self._rename_file_if_needed(
            original_name,
            only_new=getattr(settings, "DOCUMENTI_RENAME_ONLY_NEW", True),
        )

    # Gestione spostamento file
    if titolario_changed and self.file and old_percorso:
        if old_percorso != self.percorso_archivio:
            logger.info(
                "Documento id=%s: spostamento file da %s a %s",
                self.pk,
                old_percorso,
                self.percorso_archivio
            )
            self._move_file_into_archivio()
```

```
    elif self.file:
        self._move_file_into_archivio()
```

Metodo _move_file_into_archivio()

```
def _move_file_into_archivio(self):
    """
    Sposta il file nella directory di archivio corretta.
    Crea le directory se non esistono.
    """
    if not self.file:
        return

    from fascicoli.utils import ensure_archivio_path

    # Assicura che la directory esista
    ensure_archivio_path(self.percorso_archivio)

    old_path = self.file.path
    old_name = os.path.basename(old_path)

    # Calcola nuovo path relativo allo storage
    relative_dir = os.path.relpath(
        self.percorso_archivio,
        settings.ARCHIVIO_BASE_PATH
    )
    new_relative_path = os.path.join(relative_dir, old_name)

    # Usa lo storage per spostare il file
    if self.file.storage.exists(new_relative_path):
        # File già nella posizione corretta
        return

    # Leggi contenuto e salva in nuova posizione
    content = self.file.read()
    self.file.storage.delete(self.file.name)
    self.file.name = new_relative_path
    self.file.storage.save(new_relative_path, content)

    # Aggiorna il database
    type(self).objects.filter(pk=self.pk).update(file=new_relative_path)

    logger.info(f"File spostato: {old_path} -> {self.file.path}")
```

1.9 Flussi Operativi

Flusso 1: Creazione Nuovo Documento con Titolario

```
1. User upload documento con titolario "01 - Contabilità"
   ↓
2. Django salva temporaneamente in: documenti/temp/upload.pdf
   ↓
3. save() viene chiamato
   ↓
4. _build_path() costruisce:
   /mnt/archivio/rossi-spa/01-Contabilità/2025/
   ↓
5. _rename_file_if_needed() rinomina:
   upload.pdf → FAT_20250117_001.pdf
   ↓
6. _move_file_into_archivio() sposta:
   documenti/temp/FAT_20250117_001.pdf →
   rossi-spa/01-Contabilità/2025/FAT_20250117_001.pdf
   ↓
7. Risultato finale:
   /mnt/archivio/rossi-spa/01-Contabilità/2025/FAT_20250117_001.pdf
```

Flusso 2: Creazione Documento senza Titolario

```
1. User upload documento SENZA specificare titolario
   ↓
2. Django salva temporaneamente in: documenti/temp/upload.pdf
   ↓
3. save() viene chiamato
   ↓
4. _build_path() rileva titolario_voce = None
   ↓
5. get_or_create_default_titolario() ritorna "99 - Varie"
   ↓
6. Path costruito:
   /mnt/archivio/rossi-spa/99-Varie/2025/
   ↓
7. File spostato in 99-Varie
   ↓
8. Risultato:
   /mnt/archivio/rossi-spa/99-Varie/2025/TEMP_20250117_documento.pdf
```

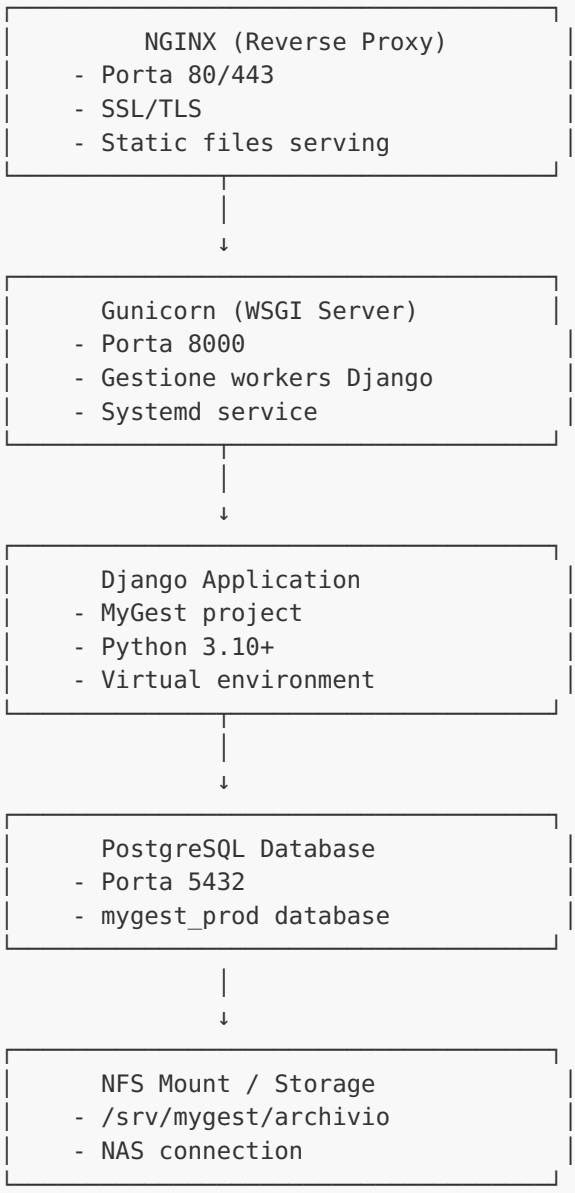
Flusso 3: Modifica Documento - Assegnazione Titolario

```
1. Documento esistente in:
   /mnt/archivio/rossi-spa/99-Varie/2025/DOC_001.pdf
   (titolario_voce = None)
   ↓
2. User modifica documento, assegna titolario "02 - Fiscale"
   ↓
3. save() viene chiamato
   ↓
4. Rileva: titolario_changed = True
   old_percorso = /mnt/archivio/rossi-spa/99-Varie/2025/
   ↓
5. _build_path() calcola nuovo percorso:
   /mnt/archivio/rossi-spa/02-Fiscale/2025/
   ↓
6. Verifica: old_percorso != self.percorso_archivio
   ↓
7. _move_file_into_archivio() sposta fisicamente:
   99-Varie/2025/DOC_001.pdf →
   02-Fiscale/2025/DOC_001.pdf
   ↓
8. Log: "Documento id=X: spostamento file da ... a ..."
   ↓
9. Risultato:
   /mnt/archivio/rossi-spa/02-Fiscale/2025/DOC_001.pdf
```

PARTE 2: DEPLOY SU VPS

2.1 Architettura Deploy

Stack Tecnologico



2.2 Preparazione VPS

2.2.1 Requisiti Sistema

```
# Sistema operativo: Ubuntu 22.04 LTS o superiore
# RAM: Minimo 2GB (consigliato 4GB+)
# Storage: Minimo 20GB + spazio per NAS mount
# CPU: 2+ cores
```

2.2.2 Update Sistema

```
# Aggiorna sistema
sudo apt update && sudo apt upgrade -y

# Installa dipendenze base
sudo apt install -y \
    python3.10 \
    python3.10-venv \
    python3-pip \
    postgresql \
    postgresql-contrib \
    nginx \
    git \
    curl \
    build-essential \
    libpq-dev \
    nfs-common
```

2.3 Configurazione PostgreSQL

2.3.1 Creazione Database

```
# Accedi a PostgreSQL
sudo -u postgres psql

# Crea database e utente
CREATE DATABASE mygest_prod;
CREATE USER mygest_user WITH PASSWORD 'secure_password_here';

# Privilegi
ALTER ROLE mygest_user SET client_encoding TO 'utf8';
ALTER ROLE mygest_user SET default_transaction_isolation TO 'read committed';
ALTER ROLE mygest_user SET timezone TO 'Europe/Rome';
GRANT ALL PRIVILEGES ON DATABASE mygest_prod TO mygest_user;

# Esci
\q
```

2.3.2 Configurazione pg_hba.conf

```
sudo nano /etc/postgresql/14/main/pg_hba.conf

# Aggiungi (prima delle altre regole):
local   mygest_prod     mygest_user                     md5
```

```
# Riavvia PostgreSQL
sudo systemctl restart postgresql
```

2.4 Setup Applicazione

2.4.1 Creazione Utente e Directory

```
# Crea utente per l'applicazione
sudo useradd -m -s /bin/bash mygest

# Crea directory
sudo mkdir -p /srv/mygest
sudo chown mygest:mygest /srv/mygest

# Passa all'utente mygest
sudo su - mygest
cd /srv/mygest
```

2.4.2 Clone Repository

```
# Clone del repository
git clone https://github.com/your-username/mygest.git app
cd app

# Oppure, se usi deploy da locale:
# (Da locale) rsync -avz --exclude='.git' /home/sandro/mygest/ user@vps:/srv/mygest/app/
```

2.4.3 Virtual Environment

```
# Crea virtual environment
python3.10 -m venv /srv/mygest/venv

# Attiva venv
source /srv/mygest/venv/bin/activate

# Aggiorna pip
pip install --upgrade pip

# Installa dipendenze
pip install -r requirements.txt
pip install gunicorn psycopg2-binary
```

2.5 Configurazione Storage NAS

2.5.1 Mount NFS

```
# Torna a utente root
exit

# Crea directory mount point
sudo mkdir -p /srv/mygest/archivio

# Configura mount NFS
sudo nano /etc/fstab

# Aggiungi:
nas-server-ip:/volume/path  /srv/mygest/archivio  nfs  defaults,_netdev  0  0

# Mount
sudo mount -a

# Verifica
df -h | grep archivio
```

2.5.2 Permessi

```
# Assegna ownership
sudo chown -R mygest:mygest /srv/mygest/archivio

# Permessi
sudo chmod -R 755 /srv/mygest/archivio

# Test scrittura
sudo -u mygest touch /srv/mygest/archivio/test.txt
ls -la /srv/mygest/archivio/test.txt
sudo -u mygest rm /srv/mygest/archivio/test.txt
```

2.6 Configurazione Django

2.6.1 File .env.production

```
sudo -u mygest nano /srv/mygest/app/.env

# Contenuto:
DATABASE_URL=postgresql://mygest_user:secure_password_here@localhost:5432/mygest_prod
ARCHIVIO_BASE_PATH=/srv/mygest/archivio
SECRET_KEY=your-very-long-and-secure-secret-key-here
DEBUG=False
ALLOWED_HOSTS=yourdomain.com,www.yourdomain.com,your-vps-ip
```

2.6.2 Migrazioni e Static Files

```
sudo su - mygest
cd /srv/mygest/app
source /srv/mygest/venv/bin/activate

# Migrazioni
python manage.py migrate

# Crea superuser
python manage.py createsuperuser

# Collect static files
python manage.py collectstatic --noinput

# Test
python manage.py check
python manage.py check --deploy
```

2.7 Configurazione Gunicorn

2.7.1 File di Configurazione

```
sudo nano /srv/mygest/app/gunicorn_config.py
```

```
# gunicorn_config.py
import multiprocessing

# Bind
bind = "127.0.0.1:8000"

# Workers
workers = multiprocessing.cpu_count() * 2 + 1
worker_class = "sync"
worker_connections = 1000
max_requests = 1000
max_requests_jitter = 50

# Timeout
timeout = 120
graceful_timeout = 30
keepalive = 5

# Logging
accesslog = "/srv/mygest/logs/gunicorn_access.log"
errorlog = "/srv/mygest/logs/gunicorn_error.log"
loglevel = "info"

# Process naming
proc_name = "mygest"

# Server mechanics
daemon = False
pidfile = "/srv/mygest/run/gunicorn.pid"
umask = 0o007

# Paths
pythonpath = "/srv/mygest/app"
chdir = "/srv/mygest/app"
```

2.7.2 Systemd Service

```
sudo nano /etc/systemd/system/mygest.service
```

```
[Unit]
Description=MyGest Gunicorn Application
After=network.target postgresql.service
Requires=postgresql.service

[Service]
Type=notify
User=mygest
Group=mygest
RuntimeDirectory=mygest
WorkingDirectory=/srv/mygest/app
Environment="PATH=/srv/mygest/venv/bin"
ExecStart=/srv/mygest/venv/bin/gunicorn \
    --config /srv/mygest/app/gunicorn_config.py \
    mygest.wsgi:application
ExecReload=/bin/kill -s HUP $MAINPID
KillMode=mixed
TimeoutStopSec=5
PrivateTmp=true
Restart=on-failure
RestartSec=5s

[Install]
WantedBy=multi-user.target
```

2.7.3 Creazione Directory Log

```
# Crea directory
sudo mkdir -p /srv/mygest/logs
sudo mkdir -p /srv/mygest/run

# Permessi
sudo chown -R mygest:mygest /srv/mygest/logs
sudo chown -R mygest:mygest /srv/mygest/run
```

2.7.4 Avvio Servizio

```
# Reload systemd
sudo systemctl daemon-reload

# Abilita servizio
sudo systemctl enable mygest

# Avvia servizio
sudo systemctl start mygest

# Verifica stato
sudo systemctl status mygest

# Log in tempo reale
sudo journalctl -u mygest -f
```

2.8 Configurazione NGINX

2.8.1 File di Configurazione

```
sudo nano /etc/nginx/sites-available/mygest
```

```
upstream mygest_app {
    server 127.0.0.1:8000;
}

server {
    listen 80;
    server_name yourdomain.com www.yourdomain.com;

    # Security headers
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header X-XSS-Protection "1; mode=block" always;

    # Client max body size (upload files)
    client_max_body_size 100M;

    # Static files
    location /static/ {
        alias /srv/mygest/app/staticfiles/;
        expires 30d;
        add_header Cache-Control "public, immutable";
    }

    # Media files (archivio)
    location /archivio/ {
        alias /srv/mygest/archivio/;

        # Security: solo utenti autenticati
        internal;

        expires 7d;
        add_header Cache-Control "private";
    }

    # Django application
    location / {
        proxy_pass http://mygest_app;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # Timeouts
        proxy_connect_timeout 60s;
        proxy_send_timeout 60s;
        proxy_read_timeout 60s;
    }

    # Logs
    access_log /var/log/nginx/mygest_access.log;
    error_log /var/log/nginx/mygest_error.log;
}
```

2.8.2 Attivazione Sito

```
# Link simbolico
sudo ln -s /etc/nginx/sites-available/mygest /etc/nginx/sites-enabled/

# Rimuovi default (opzionale)
sudo rm /etc/nginx/sites-enabled/default

# Test configurazione
sudo nginx -t

# Riavvia NGINX
sudo systemctl restart nginx

# Abilita avvio automatico
sudo systemctl enable nginx
```

2.9 SSL/TLS con Let's Encrypt

2.9.1 Installazione Certbot

```
sudo apt install -y certbot python3-certbot-nginx
```

2.9.2 Ottenimento Certificato

```
# Ottieni certificato
sudo certbot --nginx -d yourdomain.com -d www.yourdomain.com

# Segui le istruzioni:
# 1. Inserisci email
# 2. Accetta Terms of Service
# 3. Scegli redirect automatico HTTP -> HTTPS
```

2.9.3 Rinnovo Automatico

```
# Test rinnovo
sudo certbot renew --dry-run

# Il rinnovo automatico è già configurato in:
# /etc/cron.d/certbot
```

2.9.4 Configurazione NGINX Post-SSL

Certbot modifica automaticamente `/etc/nginx/sites-available/mygest`:

```
server {
    listen 80;
    server_name yourdomain.com www.yourdomain.com;
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name yourdomain.com www.yourdomain.com;

    # SSL Certificate
    ssl_certificate /etc/letsencrypt/live/yourdomain.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/yourdomain.com/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    # ... resto della configurazione ...
}
```

2.10 Backup e Restore

2.10.1 Script Backup Database

```
sudo nano /srv/mygest/scripts/backup_db.sh
```

```
#!/bin/bash
# Backup database PostgreSQL

BACKUP_DIR="/srv/mygest/backups/db"
DATE=$(date +%Y%m%d_%H%M%S)
FILENAME="mygest_backup_${DATE}.dump"

mkdir -p "$BACKUP_DIR"

# Dump database
pg_dump -U mygest_user -h localhost -F c -b -v \
    -f "$BACKUP_DIR/$FILENAME" mygest_prod

# Comprimi
gzip "$BACKUP_DIR/$FILENAME"

# Mantieni solo ultimi 7 giorni
find "$BACKUP_DIR" -type f -mtime +7 -delete

echo "Backup completato: $FILENAME.gz"
```

```
# Permessi
sudo chmod +x /srv/mygest/scripts/backup_db.sh
sudo chown mygest:mygest /srv/mygest/scripts/backup_db.sh
```

2.10.2 Cron Job Backup

```
sudo crontab -u mygest -e

# Aggiungi:
# Backup giornaliero alle 2:00 AM
0 2 * * * /srv/mygest/scripts/backup_db.sh >> /srv/mygest/logs/backup.log 2>&1
```

2.10.3 Restore Database

```
# Decomprimi backup
gunzip /srv/mygest/backups/db/mygest_backup_YYYYMMDD_HHMMSS.dump.gz

# Restore
pg_restore -U mygest_user -h localhost -d mygest_prod -v \
    -c /srv/mygest/backups/db/mygest_backup_YYYYMMDD_HHMMSS.dump
```

2.10.4 Backup File Archivio

```
# Sync archivio su backup storage
rsync -avz --delete /srv/mygest/archivio/ backup-server:/backups/mygest/archivio/

# O backup su NAS stesso (se configurato)
rsync -avz /srv/mygest/archivio/ /mnt/backup-nas/mygest/
```

2.11 Migrazione Dati da Locale a Produzione

2.11.1 Dump Database Locale

```
# Su macchina locale (WSL)
cd /home/sandro/mygest

# Dump database
pg_dump -U mygest_user -h localhost -F c -b -v \
    -f mygest_local_$(date +%Y%m%d).dump mygest
```

2.11.2 Trasferimento File

```
# Trasferisci dump
scp mygest_local_YYYYMMDD.dump user@vps:/tmp/

# Trasferisci file archivio
rsync -avz -e ssh --progress /mnt/archivio/ \
    user@vps:/srv/mygest/archivio/
```

2.11.3 Restore su VPS

```
# Su VPS
cd /srv/mygest/app

# Stop applicazione
sudo systemctl stop mygest

# Restore database
pg_restore -U mygest_user -h localhost -d mygest_prod -v \
    -c /tmp/mygest_local_YYYYMMDD.dump

# Verifica permessi archivio
sudo chown -R mygest:mygest /srv/mygest/archivio

# Riavvia applicazione
sudo systemctl start mygest
```

2.12 Monitoraggio e Manutenzione

2.12.1 Comandi Utili

```
# Stato servizi
sudo systemctl status mygest
sudo systemctl status nginx
sudo systemctl status postgresql

# Log applicazione
sudo journalctl -u mygest -f
sudo journalctl -u mygest --since "1 hour ago"

# Log NGINX
sudo tail -f /var/log/nginx/mygest_access.log
sudo tail -f /var/log/nginx/mygest_error.log

# Log Gunicorn
sudo tail -f /srv/mygest/logs/gunicorn_access.log
sudo tail -f /srv/mygest/logs/gunicorn_error.log

# Spazio disco
df -h
du -sh /srv/mygest/archivio

# Processi
ps aux | grep gunicorn
htop
```

2.12.2 Restart Applicazione

```
# Restart completo
sudo systemctl restart mygest

# Reload senza downtime (HUP signal)
sudo systemctl reload mygest

# Restart NGINX
sudo systemctl restart nginx
```

2.12.3 Update Applicazione

```
# Su VPS, come utente mygest
cd /srv/mygest/app
source /srv/mygest/venv/bin/activate

# Pull nuove modifiche
git pull origin main

# Update dipendenze
pip install -r requirements.txt

# Migrazioni
python manage.py migrate

# Collect static
python manage.py collectstatic --noinput

# Restart
sudo systemctl restart mygest
```

2.13 Troubleshooting

2.13.1 Applicazione Non Parte

```
# Verifica log systemd
sudo journalctl -u mygest -n 100 --no-pager

# Verifica configurazione
cd /srv/mygest/app
source /srv/mygest/venv/bin/activate
python manage.py check

# Verifica permessi
ls -la /srv/mygest/app
ls -la /srv/mygest/venv

# Test manuale Gunicorn
/srv/mygest/venv/bin/gunicorn \
    --config /srv/mygest/app/gunicorn_config.py \
    --bind 127.0.0.1:8000 \
    mygest.wsgi:application
```

2.13.2 Errori Database

```
# Verifica connessione PostgreSQL
sudo -u postgres psql -c "\l"
sudo -u postgres psql -d mygest_prod -c "\dt"

# Test connessione da Django
cd /srv/mygest/app
source /srv/mygest/venv/bin/activate
python manage.py dbshell

# Verifica .env
cat /srv/mygest/app/.env | grep DATABASE_URL
```

2.13.3 Problemi Storage/NAS

```
# Verifica mount
df -h | grep archivio
mount | grep archivio

# Test permessi
sudo -u mygest touch /srv/mygest/archivio/test.txt
ls -la /srv/mygest/archivio/test.txt

# Remount se necessario
sudo umount /srv/mygest/archivio
sudo mount -a
```

2.13.4 NGINX Errori

```
# Test configurazione
sudo nginx -t

# Log dettagliati
sudo tail -100 /var/log/nginx/mygest_error.log

# Verifica permessi socket/porta
sudo netstat -tulpn | grep :80
sudo netstat -tulpn | grep :443
sudo netstat -tulpn | grep :8000
```

2.14 Security Checklist

2.14.1 Checklist Pre-Produzione

- [ ] `DEBUG=False` in `.env`
- [ ] `SECRET_KEY` lunga e casuale (50+ caratteri)
- [ ] `ALLOWED_HOSTS` configurato correttamente
- [ ] PostgreSQL: password forte per utente database
- [ ] PostgreSQL: pg_hba.conf configurato per local access
- [ ] Firewall: solo porte 80, 443, 22 aperte
- [ ] SSL/TLS configurato con Let's Encrypt
- [ ] NGINX: security headers attivi
- [ ] File archivio: permessi 755, owner mygest
- [ ] Backup automatico configurato
- [ ] Monitoring configurato (opzionale: Sentry, New Relic)

2.14.2 Configurazione Firewall (UFW)

```
# Installa UFW
sudo apt install -y ufw

# Regole di base
sudo ufw default deny incoming
sudo ufw default allow outgoing

# Permetti SSH
sudo ufw allow 22/tcp

# Permetti HTTP/HTTPS
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp

# Attiva firewall
sudo ufw enable

# Verifica status
sudo ufw status verbose
```

2.15 Performance Optimization

2.15.1 PostgreSQL Tuning

```
sudo nano /etc/postgresql/14/main/postgresql.conf
```

```
# Memory Settings (per 4GB RAM)
shared_buffers = 1GB
effective_cache_size = 3GB
maintenance_work_mem = 256MB
work_mem = 16MB

# Connection Settings
max_connections = 100

# Write Ahead Log
wal_buffers = 16MB
checkpoint_completion_target = 0.9

# Query Planner
random_page_cost = 1.1  # SSD
effective_io_concurrency = 200  # SSD
```

```
sudo systemctl restart postgresql
```

2.15.2 Gunicorn Workers

```
# gunicorn_config.py
import multiprocessing

# Formula: (2 x CPU_CORES) + 1
workers = multiprocessing.cpu_count() * 2 + 1

# Per server con 2 CPU: 5 workers
# Per server con 4 CPU: 9 workers
```

2.15.3 NGINX Caching

```
# In /etc/nginx/sites-available/mygest

# Cache zone
proxy_cache_path /var/cache/nginx levels=1:2 keys_zone=mygest_cache:10m
                 max_size=100m inactive=60m use_temp_path=off;

server {
    # ...

    location / {
        proxy_cache mygest_cache;
        proxy_cache_valid 200 10m;
        proxy_cache_bypass $http_cache_control;
        add_header X-Cache-Status $upstream_cache_status;

        proxy_pass http://mygest_app;
        # ...
    }
}
```

PARTE 3: CHECKLIST E SCRIPT UTILI

3.1 Checklist Deploy Completo

Pre-Deploy

- [ ] Codice testato in locale
- [ ] Migrazioni database create e testate
- [ ] File `.env.production` preparato con credenziali corrette
- [ ] Backup database locale eseguito
- [ ] Lista dipendenze aggiornata (`requirements.txt`)

VPS Setup

- [ ] Sistema aggiornato (`apt update && upgrade`)
- [ ] PostgreSQL installato e configurato
- [ ] Database production creato
- [ ] Utente `mygest` creato
- [ ] Directory `/srv/mygest` create
- [ ] NFS/NAS mount configurato e testato
- [ ] Permessi storage verificati

Applicazione

- [ ] Repository clonato in `/srv/mygest/app`
- [ ] Virtual environment creato
- [ ] Dipendenze installate
- [ ] File `.env` configurato
- [ ] Migrazioni eseguite
- [ ] Static files collected
- [ ] Superuser creato
- [ ] `manage.py check --deploy` eseguito senza errori

Servizi

- [ ] Gunicorn configurato (`gunicorn_config.py`)
- [ ] Systemd service creato e attivo
- [ ] NGINX configurato
- [ ] NGINX attivo e funzionante
- [ ] SSL/TLS configurato (Let's Encrypt)
- [ ] Redirect HTTP -> HTTPS attivo

Security

- [ ] Firewall configurato (UFW)
- [ ] Porte non necessarie chiuse
- [ ] `DEBUG=False`
- [ ] `SECRET_KEY` secure
- [ ] Security headers NGINX attivi
- [ ] File permissions corrette

Post-Deploy

- [ ] Test accesso web funzionante
- [ ] Test login admin panel
- [ ] Test upload documento
- [ ] Test creazione cliente/fascicolo
- [ ] Verifica log senza errori
- [ ] Backup automatico configurato
- [ ] Monitoraggio configurato

3.2 Script di Deploy Automatico

3.2.1 Script deploy.sh

```
#!/bin/bash
# Script di deploy automatico MyGest

set -e  # Exit on error

echo "=== Deploy MyGest su VPS ==="
echo ""

# Variabili
APP_DIR="/srv/mygest/app"
VENV_DIR="/srv/mygest/venv"
USER="mygest"

echo "1. Pull codice da repository..."
cd "$APP_DIR"
sudo -u "$USER" git pull origin main

echo ""
echo "2. Attiva virtual environment..."
source "$VENV_DIR/bin/activate"

echo ""
echo "3. Aggiorna dipendenze..."
pip install -r requirements.txt --quiet

echo ""
echo "4. Esegui migrazioni database..."
python manage.py migrate --noinput

echo ""
echo "5. Collect static files..."
python manage.py collectstatic --noinput --clear

echo ""
echo "6. Run checks..."
python manage.py check --deploy

echo ""
echo "7. Restart applicazione..."
sudo systemctl restart mygest

echo ""
echo "8. Verifica stato servizio..."
sleep 2
sudo systemctl status mygest --no-pager -l

echo ""
echo "=== Deploy completato con successo! ==="
```

```
# Salva in /srv/mygest/scripts/deploy.sh
sudo chmod +x /srv/mygest/scripts/deploy.sh
```

3.2.2 Utilizzo

```
# Deploy da VPS
cd /srv/mygest/scripts
sudo ./deploy.sh

# Deploy da locale (via SSH)
ssh user@vps "cd /srv/mygest/scripts && sudo ./deploy.sh"
```

3.3 Script Verifica Storage

```
#!/bin/bash
# Verifica configurazione storage e permessi

echo "=== Verifica Storage MyGest ==="
echo ""

ARCHIVIO_PATH="/srv/mygest/archivio"

echo "1. Verifica mount point..."
if mount | grep -q "$ARCHIVIO_PATH"; then
    echo "✓ Storage montato correttamente"
    df -h "$ARCHIVIO_PATH"
else
    echo "✗ ERRORE: Storage non montato!"
    exit 1
fi

echo ""
echo "2. Verifica permessi..."
OWNER=$(stat -c '%U:%G' "$ARCHIVIO_PATH")
if [ "$OWNER" == "mygest:mygest" ]; then
    echo "✓ Permessi corretti: $OWNER"
else
    echo "⚠ Warning: Permessi attuali: $OWNER (atteso: mygest:mygest)"
fi

echo ""
echo "3. Test scrittura..."
TEST_FILE="$ARCHIVIO_PATH/.storage_test_$$"
if sudo -u mygest touch "$TEST_FILE" 2>/dev/null; then
    echo "✓ Test scrittura riuscito"
    sudo -u mygest rm "$TEST_FILE"
else
    echo "✗ ERRORE: Impossibile scrivere su storage!"
    exit 1
fi

echo ""
echo "4. Spazio disponibile..."
SPACE=$(df -h "$ARCHIVIO_PATH" | awk 'NR==2 {print $4}')
echo "Spazio libero: $SPACE"

echo ""
echo "5. Struttura directory..."
find "$ARCHIVIO_PATH" -maxdepth 2 -type d | head -20

echo ""
echo "=== Verifica completata ==="
```

3.4 Script Monitoring

```
#!/bin/bash
# Monitoring servizi MyGest

echo "=== Status MyGest Services ==="
echo ""

# Gunicorn
echo "1. Gunicorn (MyGest App):"
if systemctl is-active --quiet mygest; then
    echo "  Status: ✓ Running"
    echo "  Uptime: $(systemctl show mygest --property=ActiveEnterTimestamp --value)"
else
    echo "  Status: ✗ Stopped"
fi

# NGINX
echo ""
echo "2. NGINX:"
if systemctl is-active --quiet nginx; then
    echo "  Status: ✓ Running"
else
    echo "  Status: ✗ Stopped"
fi

# PostgreSQL
echo ""
echo "3. PostgreSQL:"
if systemctl is-active --quiet postgresql; then
    echo "  Status: ✓ Running"
    echo "  Connections: $(sudo -u postgres psql -t -c "SELECT count(*) FROM pg_stat_activity WH
else
    echo "  Status: ✗ Stopped"
fi

# Storage
echo ""
echo "4. Storage:"
if mount | grep -q "/srv/mygest/archivio"; then
    echo "  Status: ✓ Mounted"
    df -h /srv/mygest/archivio | awk 'NR==2 {printf "  Usage: %s / %s (%s used)\n", $3, $2, $5}'
else
    echo "  Status: ✗ Not mounted"
fi

# Resources
echo ""
echo "5. System Resources:"
echo "  CPU: $(top -bn1 | grep "Cpu(s)" | sed "s/.*, *\([0-9.]*\)%* id.*/\1/" | awk '{print 100
echo "  RAM: $(free -h | awk 'NR==2 {printf "%s / %s (%s)\n", $3, $2, $3/$2*100"%"}')"
echo "  Disk: $(df -h / | awk 'NR==2 {printf "%s / %s (%s)\n", $3, $2, $5}')"

# Recent Errors
echo ""
echo "6. Recent Errors (last 10):"
sudo journalctl -u mygest --since "1 hour ago" -p err -n 10 --no-pager | grep -v "^--" || echo "
```

```
echo ""
echo "==================================="
```

CONCLUSIONI

Riepilogo Architettura

```
Cliente → NGINX (SSL) → Gunicorn → Django
                                    ↓
                                    |
                              ┌─────┴─────┐
                              ↓           ↓
                         PostgreSQL    NAS Storage
                                    (/srv/mygest/archivio)
```

Path Storage Finale

Locale (WSL):

```
/mnt/archivio/{cliente}/{titolario}/{anno}/documento.pdf
```

Produzione (VPS):

```
/srv/mygest/archivio/{cliente}/{titolario}/{anno}/documento.pdf
```

Contatti e Supporto

- Documentazione: `/srv/mygest/app/docs/`
- Log applicazione: `/srv/mygest/logs/`
- Log sistema: `journalctl -u mygest`

Guida creata il: 17 Novembre 2025
Versione: 1.0
Sistema: MyGest - File Storage e Deploy VPS