Inhaltsverzeichnis

1. Betriebsdokumentation: StudIQ

1.1. Einleitung

Diese Dokumentation richtet sich an Administratoren und unterstützt bei der Einrichtung, Konfiguration und Betreuung des StudIQ-Systems.

StudIQ ist eine webbasierte Lernplattform zur Prüfungsvorbereitung für HTW-Studierende. Das System besteht aus:

- **Frontend**: Vue.js 3 Single-Page-Application
- **Backend**: Django 5.2 REST-API
- **Datenbank**: MySQL/MariaDB

1.2. Produktionssystem

Die aktuelle liveversion des Systems ist verfügbar unter:

https://studiq.oceanbound.io/

1.3. Systemvoraussetzungen

1.3.1. Hardware-Mindestanforderungen

Komponente	Minimum	Empfohlen
CPU	2 Kerne	4 Kerne
RAM	2 GB	4 GB
Festplatte	10 GB	20 GB SSD
Netzwerk	100 Mbit/s	1 Gbit/s

1.3.2. Softwareanforderungen

Server

Software	Version	Hinweise
Betriebssystem	Linux (Ubuntu 22.04+, Debian 11+) oder Windows Server 2019+	Linux empfohlen
Python	3.10+	Python Downloads
Node.js	18.x LTS oder 20.x LTS	Node.js Downloads
MySQL/MariaDB	MySQL 8.0+ / MariaDB 10.6+	MySQL Dokumentation

Client (Browser)

Browser	Mindestversion
Google Chrome	100+
Mozilla Firefox	100+
Microsoft Edge	100+
Safari	15+

JavaScript muss aktiviert sein. Cookies müssen für die Authentifizierung erlaubt sein.

1.4. Systemeinrichtung

1.4.1. Quellcode beziehen

Das Repository klonen:

```
git clone https://github.com/Gorg-tech/StudIQ studiq
```

```
cd studiq
```

1.4.2. Backend-Einrichtung

MySQL/MariaDB installieren und konfigurieren

Django erstellt **keine** Datenbank automatisch. MySQL/MariaDB muss manuell installiert und die Datenbank samt Benutzer erstellt werden.

Installation (Ubuntu/Debian):

```
sudo apt update
sudo apt install mysql-server
sudo mysql_secure_installation  # Empfohlen: Root-Passwort setzen, Test-DB entfernen
```

Weitere Informationen zur MySQL-Installation finden Sie in der offiziellen MySQL-Dokumentation.

Datenbank und Benutzer erstellen:

```
# Als MySQL-Root-Benutzer anmelden
sudo mysql -u root -p

# In der MySQL-Konsole:
CREATE DATABASE studiq_db CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
CREATE USER 'studiq_user'@'localhost' IDENTIFIED BY 'IHR_SICHERES_PASSWORT';
GRANT ALL PRIVILEGES ON studiq_db.* TO 'studiq_user'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

Ersetzen Sie `IHR_SICHERES_PASSWORT` durch ein sicheres Passwort und verwenden Sie dasselbe Passwort in der `.env`-Datei!

Verbindung testen:

```
mysql -u studiq_user -p studiq_db
# Passwort eingeben - sollte ohne Fehler verbinden
```

1.4.3. Python-Umgebung erstellen

```
cd src/server
python3 -m venv venv
source venv/bin/activate  # Linux/macOS
# oder: venv\Scripts\activate  # Windows
```

```
pip install -r requirements.txt
```

Umgebungsvariablen konfigurieren

Erstellen Sie eine `.env`-Datei im Verzeichnis `src/server/`:

```
cp .env.sample .env
```

Bearbeiten Sie die `.env`-Datei mit den korrekten Werten:

```
# Datenbank-Konfiguration
DB_ENGINE=django.db.backends.mysql
DB_NAME=studiq_db
DB_USER=studiq_user
DB_PASSWORD=<IHR_SICHERES_PASSWORT>
DB_HOST=<DATENBANK_HOST>
DB_PORT=3306

# Django-Konfiguration (für Produktion setzen!)
DJANGO_SECRET_KEY=<ZUFÄLLIGER_SICHERER_SCHLÜSSEL>
DJANGO_DEBUG=False
DJANGO_ALLOWED_HOSTS=ihre-domain.de,www.ihre-domain.de
```

Für Produktion unbedingt einen neuen `DJANGO_SECRET_KEY` generieren und `DJANGO_DEBUG=False` setzen!

Tabelle 1. Umgebungsvariablen im Detail

Variable	Beschreibung	Standardwert
`DB_ENGINE`	Datenbank-Engine	`django.db.backends.mysql`
`DB_NAME`	Name der Datenbank	`studiq_db`
`DB_USER`	Datenbank-Benutzer	`studiq_user`
`DB_PASSWORD`	Datenbank-Passwort	*Muss gesetzt werden*
`DB_HOST`	IP/Hostname des Datenbankservers (bei DB auf gleichem Server: `localhost` oder `127.0.0.1`)	*Muss gesetzt werden*
`DB_PORT`	Port des Datenbankservers	`3306`
`DJANGO_SECRET_KEY`	Geheimer Schlüssel für Django	*Entwicklungs-Default*
`DJANGO_DEBUG`	Debug-Modus aktivieren (für Produktion: `False`)	`True`
`DJANGO_ALLOWED_HOSTS`	Erlaubte Hosts (kommagetrennt)	*leer*

Datenbank initialisieren

```
# Datenbank-Migrationen ausführen
python3 manage.py migrate

# Initiale Daten laden (Studiengänge, Module etc. aus bereitgestellten JSON-Dateien)
python3 scraper/populate_db.py

# Admin-Benutzer erstellen
python3 manage.py createsuperuser
```

Die initialen Daten sind bereits als JSON-Dateien im Repository enthalten. Um aktuelle Daten von Modulux zu laden, muss zuerst `modulux_scraper.py` ausgeführt werden, danach `populate_db.py` um die Datenbank zu aktualisieren.

Backend starten

Entwicklungsserver:

```
python3 manage.py runserver 0.0.0.0:8000
```

Für Produktion wird ein WSGI-Server wie Gunicorn empfohlen.

Da Gunicorn keine statischen Dateien (CSS, Bilder, JavaScript für Admin-Panel) ausliefert, wird in diesem Projekt **WhiteNoise** verwendet. Dies ermöglicht es der Python-Anwendung, ihre eigenen statischen Dateien effizient auszuliefern. WhiteNoise ist in der `requirements.txt` enthalten und in den `settings.py` bereits konfiguriert.

Um den Server im Hintergrund auszuführen (Daemon-Modus):

```
pip install gunicorn
gunicorn config.wsgi:application --bind 0.0.0.0:8000 --daemon
```

1.4.4. Frontend-Einrichtung

Abhängigkeiten installieren

```
cd src/client
npm install
```

Entwicklungsserver starten

```
npm run dev
```

Das Frontend ist dann erreichbar unter: http://localhost:5173

Produktions-Build erstellen

```
npm run build
```

Die kompilierten Dateien befinden sich in `src/client/dist/` und können von einem Webserver (z.B. Nginx) ausgeliefert werden.

1.4.5. Webserver-Konfiguration (Produktion)

Für den Produktionsbetrieb wird empfohlen, einen Reverse-Proxy wie Nginx zu verwenden.

Nginx-Beispielkonfiguration

Erstellen Sie eine Konfigurationsdatei `/etc/nginx/sites-available/studiq`:

```
server {
    listen 80;
    server_name ihre-domain.de;  # Beispiel für StudIQ-Produktionssystem:
studiq.oceanbound.io

    # Weiterleitung zu HTTPS
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name ihre-domain.de;  # Beispiel für StudIQ-Produktionssystem:
studiq.oceanbound.io

    # SSL-Zertifikate (z.B. von Let's Encrypt)
    ssl_certificate /etc/letsencrypt/live/ihre-domain.de/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/ihre-domain.de/privkey.pem;

    # Frontend (statische Dateien)
    location / {
        root /var/www/studiq/client/dist;
        try_files $uri $uri/ /index.html;
    }

    # Backend API
    location /api/ {
        proxy_pass http://127.0.0.1:8000;
        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;
    }

    # Django Admin
    location /admin/ {
        proxy_pass http://127.0.0.1:8000;
        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;
    }

    # Statische Django-Dateien
    location /static/ {
        proxy_pass http://127.0.0.1:8000;
    }
}
```

Aktivieren Sie die Konfiguration:

```
sudo ln -s /etc/nginx/sites-available/studiq /etc/nginx/sites-enabled/
sudo nginx -t  # Konfiguration testen
sudo systemctl reload nginx
```

Weitere Informationen zur Nginx-Konfiguration finden Sie in der offiziellen Nginx-Dokumentation.

SSL-Zertifikate mit Let's Encrypt

```
sudo apt install certbot python3-certbot-nginx
sudo certbot --nginx -d ihre-domain.de  # Beispiel: studiq.oceanbound.io
```

Weitere Informationen: Certbot-Dokumentation

1.4.6. CORS-Konfiguration

Für den Produktionsbetrieb müssen die CORS-Einstellungen in `src/server/config/settings.py` angepasst werden:

```
# Ersetzen Sie CORS_ALLOW_ALL_ORIGINS = True durch:
CORS_ALLOWED_ORIGINS = [
    "https://ihre-domain.de",
    "https://www.ihre-domain.de",
]

# Aktualisieren Sie CSRF_TRUSTED_ORIGINS:
```

```
CSRF_TRUSTED_ORIGINS = [
    "https://ihre-domain.de",
    "https://www.ihre-domain.de",
]
```

1.4.7. Browser-Einstellungen

Für die korrekte Nutzung von StudIQ müssen folgende Browser-Einstellungen aktiv sein:

- **JavaScript**: Aktiviert
- **Cookies**: Erlaubt für die Domain des Systems
- **Third-Party-Cookies**: Bei Cross-Domain-Setups ggf. erlauben

1.5. Systembetreuung

1.5.1. Admin-Panel

Das Django Admin-Panel ist erreichbar unter:

- Entwicklung: http://127.0.0.1:8000/admin/
- Produktion: https://ihre-domain.de/admin/

Hier können verwaltet werden:

- Benutzerkonten
- Quizze und Lernsets
- Studiengänge und Module
- Freundschaftsanfragen

1.5.2. FAQ für Benutzersupport

1. *Benutzer kann sich nicht anmelden*
    - Prüfen Sie, ob Cookies im Browser aktiviert sind
    - Prüfen Sie, ob der Benutzer existiert (Admin-Panel)
    - Passwort zurücksetzen lassen
2. *Quiz wird nicht geladen*
    - Browser-Cache leeren
    - JavaScript-Konsole auf Fehler prüfen
    - Backend-Logs auf Fehlermeldungen prüfen
3. *Daten werden nicht synchronisiert*
    - Netzwerkverbindung prüfen
    - CORS-Einstellungen überprüfen

- Browser-Entwicklertools → Netzwerk-Tab prüfen

1.5.3. Fehlerdiagnose

Log-Dateien

Django-Logs werden standardmäßig auf der Konsole ausgegeben. Für Produktion empfiehlt sich die Konfiguration einer Log-Datei:

```
# In settings.py hinzufügen:
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'handlers': {
        'file': {
            'level': 'WARNING',
            'class': 'logging.FileHandler',
            'filename': '/var/log/studiq/django.log',
        },
    },
    'loggers': {
        'django': {
            'handlers': ['file'],
            'level': 'WARNING',
            'propagate': True,
        },
    },
}
```

Test-Logs

Testausführungen werden protokolliert in: `src/server/test_logs/`

Format: `test_results_<typ>_<modul>_<datum>.txt`

Typische Fehlermeldungen

Fehler	Lösung
`OperationalError: (2003, "Can't connect to MySQL server")`	Datenbank-Verbindung prüfen (Host, Port, Firewall)
`DisallowedHost`	`DJANGO_ALLOWED_HOSTS` in `.env` konfigurieren
`CSRF verification failed`	`CSRF_TRUSTED_ORIGINS` in `settings.py` aktualisieren
`ModuleNotFoundError`	`pip install -r requirements.txt` ausführen

1.5.4. Datensicherung

Datenbank-Backup

MySQL/MariaDB:

```
mysqldump -u studiq_user -p studiq_db > backup_$(date +%Y%m%d).sql
```

Datenbank-Wiederherstellung

```
mysql -u studiq_user -p studiq_db < backup_YYYYMMDD.sql
```

Automatisierte Backups

Beispiel Cron-Job für tägliche Backups (Linux):

```
# crontab -e
0 2 * * * mysqldump -u studiq_user -p'PASSWORT' studiq_db > /backup/studiq_$(date
+\%Y\%m\%d).sql
```

> Bewahren Sie Backups an einem sicheren, vom Server getrennten Ort auf.

1.5.5. Wartungsarbeiten

Datenbank-Migrationen nach Updates

Nach einem Update des Quellcodes:

```
cd src/server
source venv/bin/activate
python3 manage.py migrate
```

Statische Dateien sammeln (Produktion)

```
python3 manage.py collectstatic
```

1.6. Weiterführende Dokumentation

- Django Dokumentation
- Vue.js Dokumentation
- MySQL Dokumentation
- Gunicorn Dokumentation
- Nginx Dokumentation

- Docker Dokumentation
- Let’s Encrypt Dokumentation