Zaawansowane Aplikacje Internetowe

- Framework Django
- Podstawowa aplikacja
- Szablony i Moduły
- Dostęp do bazy danych
- Wdrożenie projektu

Opracował: inż. Grzegorz Petri

Zagadnienia

- Migracje
- Administrator
- Panel zarządzania
- Baza danych
- Aplikacja w panelu

Migracje w Django

Migracje

- **Są sposobem frameworka do propagowania zmian (*schematów Bazy danych*) dokonywanych w modelach danych**
- **Można je traktować jak system kontroli wersji dla schematów Bazy**
- **Zmiana modelu polega np. na:**
  - dodawaniu pól w tabeli
  - usuwaniu modeli
- **Migracje są zazwyczaj automatyczne – czasem do ręcznego wykonania**

```
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work
properly until you apply the migrations for app(s): admin, auth,
contenttypes, sessions.
Run 'python manage.py migrate' to apply them.

January 1, 2022 – 00:00:00
Django version 3.2.11, using settings 'MyApp.settings'
Starting development server at http://127.0.1.1:8080/
Quit the server with CONTROL-C.
```

Polecenia migracji

Migracje

- **`makemigrations`** – tworzy nowe migracje bazujące na zmianach dokonanych na modelach
  - pakuje zmiany modelu do indywidualnych plików migracji (*git commit*)
- **`migrate`** – zatwierdza i wycofuje migracje
  - zatwierdza migracje na bazie danych (*git push*)
- **`sqlmigrate`** – wyświetla instrukcje SQL dokonujące migracji
- **`showmigrations`** – wyświetla listę migracji projektu i ich status

Polecenia migracji

Migracje

- **Zatwierdzanie wykrytych migracji ▼:**

```
$ python manage.py migrate
```

- **Przykład tworzenia pliku migracji ▼:**

```
$ python manage.py makemigrations
Migrations for 'books':
  books/migrations/0003_auto.py:
    - Alter field author on book
```

- **Migracja z etykietą:**

```
$ python manage.py makemigrations
 --name zmiana_modelu MyApp_label
```

```
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

Wsparcie backendu

Migracje

- **Migracje są wspierane na wszystkich backendach dostarczanych wraz z Django** *(z pewnymi zastrzeżeniami)*
- **Jedne bazy danych wspierają migracje lepiej, inne gorzej**
  - **PostgreSQL** – posiada najlepsze wsparcie, jednak...
  - **MySQL** – jako najpopularniejsza baza danych ma dość dużo ALE...
  - **SQLite** – jako prosta i lekka biblioteka, do mniej skomplikowanych aplikacji, posiada mało wbudowanych funkcji do modyfikacji schematów Bazy danych, WIĘC...

Wsparcie backendu

Migracje

- **PostgreSQL –** *przed wersją 11, dodawanie kolumny z domyślną wartością powoduje przepisanie całej tabeli (chyba, że zastosujemy* **null=True***)*
- **MySQL –** *brak Transakcji dla operacji zmian Schematów (*ręczna naprawa);*
  - *dokonuje pełnego przepisania tabel podczas zmian schematów;*
  - *proporcjonalnie do wielkości tabeli – witryna może zostać zablokowana*
- **SQLite –** *Django emuluje zmiany schematów (więc może to być wolniejsze) poprzez własne implementacje funkcji:*
  - *tworzenia nowych tabel z nowymi schematami*
  - *kopiowania danych pomiędzy tabelami*
  - *kasowania starych tabel*
  - *poprawiania nazw nowych tabel by odpowiadały oryginałom*

Dostęp do panelu

Administrator

- **Tworzenie (pierwszego) użytkownika administrującego wymaga zazwyczaj:**

1) Wykonanie migracji

```
$ python manage.py migrate
```

2) Stworzenie administratora

```
$ python manage.py createsuperuser
```

Dostęp do panelu

Administrator

- **Tworzenie użytkownika administrującego**

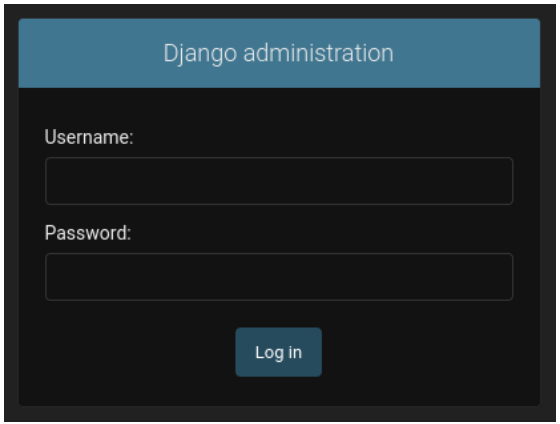
```
$ python manage.py createsuperuser
```

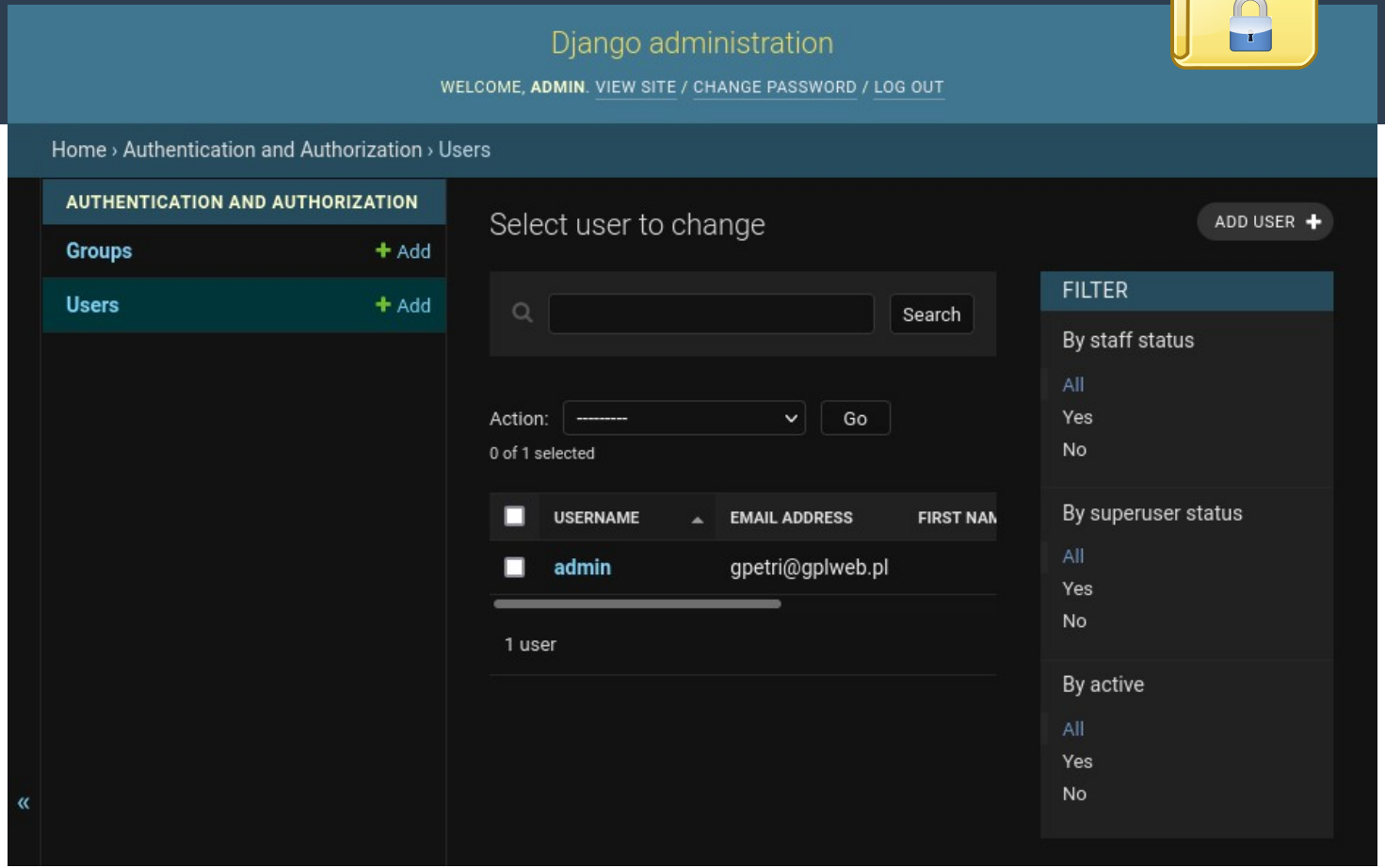
- **Kroki:**
- *Użytkownik*
- *Adres e-mail*
- *Hasło*
- *Powtórzyć*
- *Potwierdzić*

```
Username (leave blank to use 'gp'): admin
Email address: gpetri@gplweb.pl
Password:             admin
Password (again):     admin
The password is too similar to the username.
This password is too short. It must contain at least 8
characters.
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

Wygląd panelu

Panel zarządzania





- **Dostęp do panelu z adresu URL:**

`http://127.0.0.1:8000/admin/`

Schemat bazy danych

db.sqlite3

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Create Table | Create Index | Modify Table | Delete Table

Name	Type	Schema
Tables (11)		
auth_group		CREATE TABLE "auth_group" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "name" varchar(150) NOT NULL UNIQUE)
auth_group_permissions		CREATE TABLE "auth_group_permissions" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "group_id" integer NOT NULL REFERENCES
auth_permission		CREATE TABLE "auth_permission" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "content_type_id" integer NOT NULL REFERENCES "
auth_user		CREATE TABLE "auth_user" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "password" varchar(128) NOT NULL, "last_login" datetime
id	integer	`id` integer NOT NULL PRIMARY KEY AUTOINCREMENT
password	varchar (12...	`password` varchar (128) NOT NULL
last_login	datetime	`last_login` datetime
is_superuser	bool	`is_superuser` bool NOT NULL
username	varchar (15...	`username` varchar (150) NOT NULL UNIQUE
last_name	varchar (15...	`last_name` varchar (150) NOT NULL
email	varchar (25...	`email` varchar (254) NOT NULL
is_staff	bool	`is_staff` bool NOT NULL
is_active	bool	`is_active` bool NOT NULL
date_joined	datetime	`date_joined` datetime NOT NULL
first_name	varchar (15...	`first_name` varchar (150) NOT NULL
auth_user_groups		CREATE TABLE "auth_user_groups" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "user_id" integer NOT NULL REFERENCES "auth_us
auth_user_user_permissions		CREATE TABLE "auth_user_user_permissions" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "user_id" integer NOT NULL REFERENCE
django_admin_log		CREATE TABLE "django_admin_log" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "action_time" datetime NOT NULL, "object_id" text
django_content_type		CREATE TABLE "django_content_type" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "app_label" varchar(100) NOT NULL, "model" va
django_migrations		CREATE TABLE "django_migrations" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "app" varchar(255) NOT NULL, "name" varchar(255
django_session		CREATE TABLE "django_session" ("session_key" varchar(40) NOT NULL PRIMARY KEY, "session_data" text NOT NULL, "expire_date" datetime NO
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
Indices (15)		
auth_group_permissions_group_i...		CREATE INDEX "auth_group_permissions_group_id_b120cbf9" ON "auth_group_permissions" ("group_id")
auth_group_permissions_group_i...		CREATE UNIQUE INDEX "auth_group_permissions_group_id_permission_id_0cd325b0_uniq" ON "auth_group_permissions" ("group_id", "permi
auth_group_permissions_permiss...		CREATE INDEX "auth_group_permissions_permission_id_84c5c92e" ON "auth_group_permissions" ("permission_id")
auth_permission_content_type_i...		CREATE INDEX "auth_permission_content_type_id_2f476e4b" ON "auth_permission" ("content_type_id")
auth_permission_content_type_i...		CREATE UNIQUE INDEX "auth_permission_content_type_id_codename_01ab375a_uniq" ON "auth_permission" ("content_type_id", "codename"
auth_user_groups_group_id_975...		CREATE INDEX "auth_user_groups_group_id_97559544" ON "auth_user_groups" ("group_id")
auth_user_groups_user_id_6a12...		CREATE INDEX "auth_user_groups_user_id_6a12ed8b" ON "auth_user_groups" ("user_id")

Rejestracja aplikacji

Aplikacja w panelu

- **Zakładając, że nazwa aplikacji to: MyApp**
- **Otwórz/utwórz plik:**
  - ➔ `MyApp/admin.py`
- **Dopisz do niego kod ►**

MyApp/admin.py

```
from django.contrib import admin

from .models import MyApp
#   ▲ spacja
admin.site.register(MyApp)
```

```
def questions():
return answer
```