

# *DJANGO*

**SYSTEM UWIERZYTELNIANIA I  
AUTORYZACJI**

# PLAN

- Wstęp
- Uwierzytelnienie
  - Wbudowany model użytkownika Django
  - Wbudowane formularze systemu uwierzytelniania
  - Wbudowane widoki systemu uwierzytelniania
  - Wbudowane endpointy systemu uwierzytelniania
  - Udzielanie dostępu tylko uwierzytelnionemu użytkownikowi
  - User vs Admin
- Autoryzacja
  - Role
  - Uprawnienia
  - Grupy

# III ETAPY UZYSKIWANIA DOSTĘPU DO ZASOBU CHRONIONEGO

**Identyfikacja** (*ang. identification*) – zadeklarowanie tożsamości (najczęściej poprzez podanie nazwy użytkownika/maila)

**Uwierzytelnienie** (*ang. authentication*) – proces weryfikacji tożsamości (najczęściej poprzez podanie hasła/kodu przesłanego smsem/mailem)

**Autoryzacja** (*ang. authorization*) - proces weryfikacji uprawnień. Uwierzytelniony użytkownik ma zazwyczaj ograniczone uprawnienia. W szczególności może nie być uprawniony do uzyskania dostępu w żądanym zakresie.

Wszystkie trzy etapy realizowane są w Django za pomocą frameworku `django.contrib.auth`

# *UWIERZYTELNIE*

*DJANGO.CONTRIB.AUTH*



# *WBUDOWANY MODEL UZYTKOWNIKA*

KONFIGURACJA

# USTAWIENIA

Navigator Help pythonProject8 - settings.py

Add Configuration...

settings.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'relationships',
    'user_auth',
]

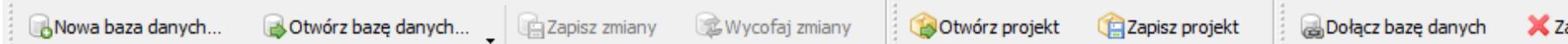
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

Projects\pythonProject8>

File

Terminal

Python Console



Struktura danych Przeglądarka danych Polecenia Pragmy Polecenia SQL

Utwórz tabelę... Utwórz indeks... Drukuj

Nazwa Rodzaj Polecenie tworzące

## ▼ Tabele (13)

- > auth\_group
- > auth\_group\_permissions
- > auth\_permission

## ▼ auth\_user

id	integer	"id" integer NOT NULL
password	varchar(128)	"password" varchar(128) NOT NULL
last_login	datetime	"last_login" datetime
is_superuser	bool	"is_superuser" bool NOT NULL
username	varchar(150)	"username" varchar(150) NOT NULL UNIQUE
last_name	varchar(150)	"last_name" varchar(150) NOT NULL
email	varchar(254)	"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(150)	"first_name" varchar(150) NOT NULL

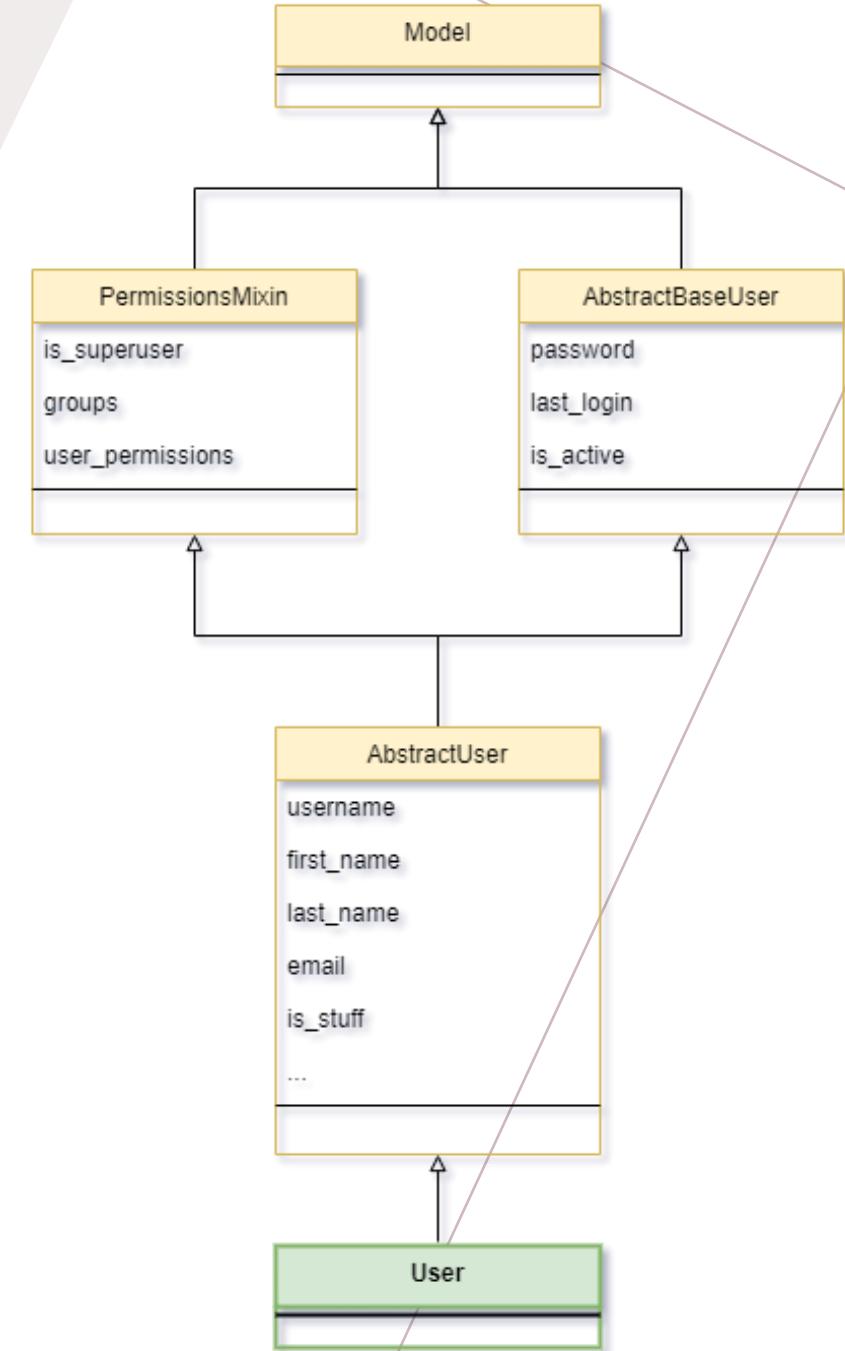
- > auth\_user\_groups
- > auth\_user\_user\_permissions
- > django\_admin\_log
- > django\_content\_type
- > django\_migrations
- > django\_session
- > relationships\_capital
- > relationships\_country
- > sqlite\_sequence

*TABELA USER*

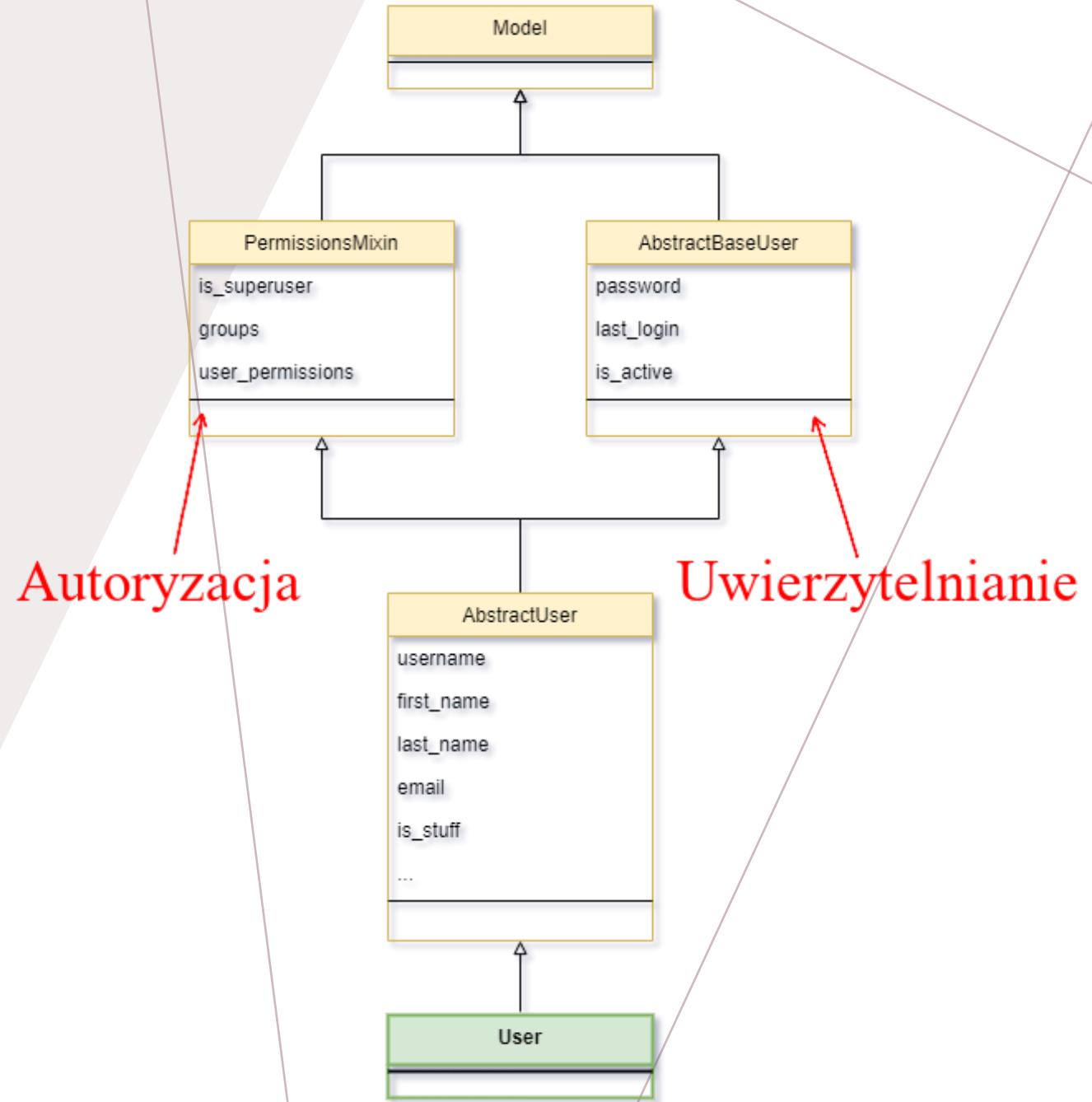
# *ATRYBUTY WBUDOWANEGO MODELU UZYTKOWNIKA*

username	password	first_name	last_name
email	is_staff	is_active	date_joined
last_login	is_superuser	groups	user_permissions

# *STRUKTURA WBUDOWANEGO MODELU UZYTKOWNIKA*



# *STRUKTURA WBUDOWANEGO MODELU UZYTKOWNIKA*



File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help auth\_proj - C:\Users\jerem\PycharmProjects\auth\_proj\venv\Lib\site-packages\django\contrib\auth\models.py

\_proj > venv > Lib > site-packages > django > contrib > auth > models.py main Reader Mode

Project auth\_proj C:\Users\jerem\PycharmProject

accounts

migrations

\_\_init\_\_.py  
admin.py  
apps.py  
models.py  
tests.py  
views.py

config

\_\_init\_\_.py  
asgi.py  
settings.py  
urls.py  
wsgi.py

venv library root

db.sqlite3  
manage.py

External Libraries

Scratches and Consoles

models.py x

```
400
401
402 class User(AbstractUser):
403     """
404     Users within the Django authentication system are represented by this
405     model.
406
407     Username and password are required. Other fields are optional.
408     """
409
410     class Meta(AbstractUser.Meta):
411         swappable = "AUTH_USER_MODEL"
412
413
414     class AnonymousUser(AbstractUser):
```

*IMPLEMENTACJA WBUDOWANEGO MODELU UZYTKOWNIKA*

*MODUL DJANGO.CONTRIB.AUTH.MODELS*

# *IMPLEMENTACJA WBUDOWANEGO MODELU UZYTKOWNIKA*

Wbudowany model użytkownika (User) to pusta implementacja modelu AbstractUser (patrz poprzedni slajd). Oznacza to, że wszystkie atrybuty i metody zaimplementowane są w klasie AbstractUser. Dzięki temu w łatwy sposób możemy stworzyć własną implementację klasy AbstractUser, a następnie dodać do niej nowe atrybuty i metody (patrz rozdział Wbudowany model użytkownika - sposoby rozszerzania).

Site administration | Django site adm X +

← → C 127.0.0.1:8000/admin/ 120% ☆

# Django administration

WELCOME, ADMIN. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

## Site administration

AUTHENTICATION AND AUTHORIZATION

	+ Add	Change
Groups		
Users		

Recent actions

My actions

None available

*REPREZENTACJA W PANELU ADMINISTRATORA*

# Django administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Accounts > Users

Select user to change

ADD USER +

Search

Action: ----- Go 0 of 3 selected

<input type="checkbox"/>	USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
<input type="checkbox"/>	Bill				<span style="color: red;">✖</span>
<input type="checkbox"/>	Sam				<span style="color: red;">✖</span>
<input type="checkbox"/>	admin	admin@admin.com			<span style="color: green;">✓</span>

3 users

## FILTER

By staff status

All

Yes

No

By superuser status

All

Yes

No

By active

All

Yes

No

*REPREZENTACJA W PANELU ADMINISTRATORA*

*WIDOK LISTY*

# *REPREZENTACJA W PANELU ADMINISTRATORA*

WIDOK SZCZEGOLU



# *WBUDOWANY MODEL UZYTKOWNIKA*

INFORMACJE PODSTAWOWE

# 1. WBUDOWANY MODEL UZYTKOWNIKA MA 20 LAT

Model użytkownika i system uwierzytelniania to podstawowy komponent wszystkich portali. Przez wzgląd na utrzymanie wstępnej kompatybilności twórcy Django do tej pory nie wprowadzili wielu poprawek do wbudowanego modelu użytkownika. W zamian Django udostępnia wiele sposobów na zmodyfikowanie tego modelu. Należy jednak pamiętać, że niektóre ze zmian powinny zostać wykonane na samym początku projektu. W ten sposób możemy uniknąć trudności na późniejszych etapach rozwoju.

## 2. PODMIENIENIE WBUDOWANEGO MODELU UZYTKOWNIKA JEST ZNACZNIE TRUDNIEJSZE NA POZNIEJSZYCH ETAPACH PROJEKTU

Podmiana wbudowanego modelu użytkownika jest znacznie trudniejsza, kiedy projekt "pracuje" już na produkcji (tj. zawiera rzeczywiste dane). W takiej sytuacji trzeba będzie przenieść już istniejące dane do nowej tabeli, a zmiana wartości zmiennej AUTH\_USER\_MODEL wymagać będzie ręcznej modyfikacji wszystkich istniejące powiązań ze starym modelem użytkownika.



# *WBUDOWANY MODEL UZYTKOWNIKA*

SPOSÓB UŻYCIA

# I. W PROCESIE UWIERZYTELNIANIA ZACHOWAJ JEDEN MODEL UZYTKOWNIKA

Zawsze staraj się zachować w swoim projekcie jeden i tylko jeden model użytkownika, w którym przechowysz będziesz wszystkie niezbędne w procesie uwierzytelniania użytkowników dane. Myśl o tym modelu w kategoriach konta, a nie użytkownika (wszyscy użytkownicy potrzebują konta, żeby się zalogować). Wszystkie konta będą współdzielić logikę typu: logowanie, wylogowanie, reset hasła, zmiana hasła, ...

Jeżeli przewidujesz w projekcie kilka typów użytkowników (np. nauczyciel i uczeń, podwładny i przełożony) możesz je wprowadzić za pomocą oddzielnej kolumny lub tabeli rozszerzającej.

## 2. NIGDY NIE UZYWAJ BEZPOSREDNIO WBUDOWANEGO MODELU UZYTKOWNIKA

Nawet jeżeli wbudowany model użytkownika spełnia wszystkie założenia Twojego projektu to przynajmniej rozszerz model `AbstractUser` pustą klasą i przypisz ją do zmiennej `AUTH_USER_MODEL` w `settings.py` (patrz następny slajd).

Takie rozwiązanie pozwoli Ci na łatwe rozszerzanie modelu, gdyby w przyszłości pojawiła się potrzeba wprowadzenia zmian w modelu. Podmiana wbudowanego modelu użytkownika na inny będzie znacznie trudniejsza, jeżeli tego nie zrobisz przed rozpoczęciem zasilania bazy rzeczywistymi danymi. Trzeba będzie przenieść już istniejące dane do nowej tabeli, a zmiana wartości zmiennej `AUTH_USER_MODEL` wymagać będzie ręcznej modyfikacji wszystkich istniejące powiązań ze starym modelem użytkownika. W przypadku relacji wiele do wielu może się to okazać szczególnie bolesne.

\_proj &gt; config &gt; settings.py

main

The screenshot shows the PyCharm IDE interface with two code editors open:

- models.py** (Left Editor):

```
1 from django.db import models
2 from django.contrib.auth.models import AbstractUser
3
4 # Create your models here.
5
6 class User(AbstractUser):
7     pass
```

A red box highlights the class definition.
- settings.py** (Right Editor):

```
121 # Default primary key field type
122 # https://docs.djangoproject.com/en/4.0/ref/sett...
123
124 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoFi...
125
126 AUTH_USER_MODEL = 'accounts.User'
127
```

The project structure on the left shows a directory tree for the 'auth\_proj' project, including 'accounts', 'migrations', and 'config' subfolders. The 'config' folder contains files like \_\_init\_\_.py, asgi.py, settings.py, urls.py, and wsgi.py. The 'venv' library root is also listed.

terminal: Local +

```
Applying admin.0003_logentry_add_action_flag_choices... OK
Applying sessions.0001_initial... OK
```

*NADPISANIE DOMYSLNEGO MODELU  
UZYTKOWNIKA*

## AUTH\_USER\_MODEL

Zmienna AUTH\_USER\_MODEL przyjmuje wartość zgodnie z konwencją:

$$AUTH\_USER\_MODEL = '<app\_name>. <model\_name>'$$

### 3. IMPORT WŁASNEGO MODELU UZYTKOWNIKA

Po nadpisaniu wbudowanego modelu użytkownika nie importuj swojego modelu użytkownika bezpośrednio. Używaj do tego funkcji `get_user_model` z modułu `django.contrib.auth`. Funkcja zwraca ustawiony w projekcie model użytkownika (na podstawie wartości zmiennej `AUTH_USER_MODEL`).

```
from django.contrib.auth import get_user_model  
  
User = get_user_model()
```



# *WBUDOWANY MODEL UZYTKOWNIKA*

SPOSÓBY ROZSZERZANIA

Istnieją cztery sposoby rozszerzenia wbudowanego modelu User:

1. Z użyciem modelu Proxy
2. Poprzez rozszerzenie modelu AbstractUser
3. Za pomocą powiązania jeden-do-jednego z modelem rozszerzającym
4. Poprzez rozszerzenie modelu AbstractBaseUser

# *MODEL PROXY*

# *KIEDY STOSOWAC?*

Kiedy nie potrzebujemy dodawać do wbudowanego modelu użytkownika żadnych dodatkowych informacji. Chcemy jedynie uzupełnić model o jakieś metody.

# MODEL PROXY

File Navigate Code Refactor Run Tools VCS Window DB Navigator Help pythonProject8 - models.py

/ user\_auth > models.py

pythonProject8 C:\Users\jerem\PycharmProjects\pythonProject8  
intro  
relationships  
user\_auth  
> migrations  
  \_\_init\_\_.py  
  admin.py  
  apps.py  
  models.py  
  tests.py  
  urls.py  
  views.py

> venv library root  
  db.sqlite3  
  manage.py  
> External Libraries  
Scatches and Consoles

Terminal: Local +

Microsoft Windows [Version 10.0.19042.1348]

(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

(venv) C:\Users\jerem\PycharmProjects\pythonProject8>

from django.contrib.auth.models import User

class Person(User):

  class Meta:

    proxy = True

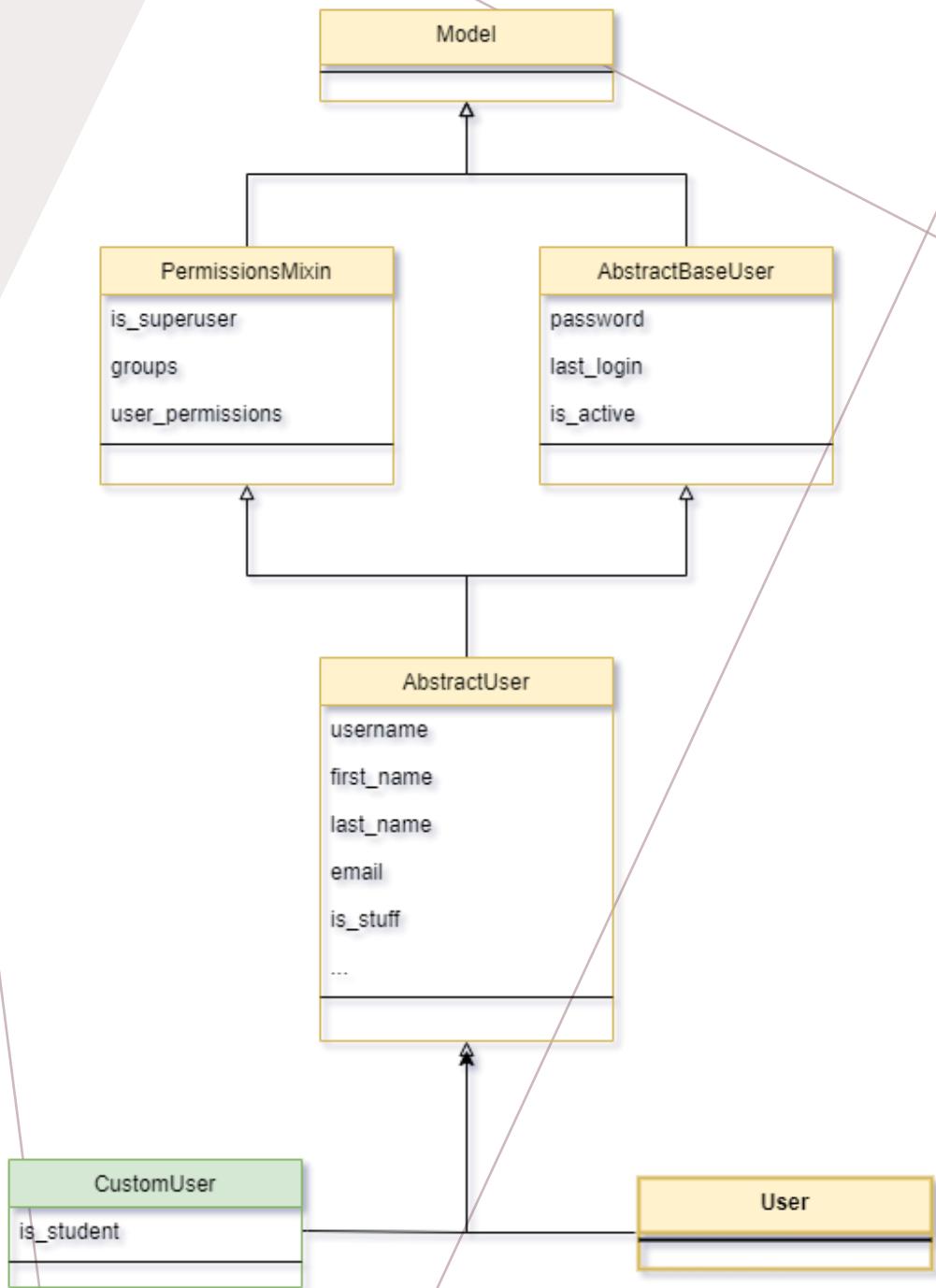
  def do\_something(self):

    ...



# *ROZSzerzenie modelu ABSTRACTUSER*

# *ROZSZERZENIE MODELU ABSTRACTUSER*



# *KIEDY STOSOWAC?*

Kiedy potrzebujemy uzupełnić wbudowany model użytkownika o dodatkowe informacje podawane w procesie rejestracji.

# ROZSZERZENIE MODELU ABSTRACTUSER

The screenshot shows a PyCharm IDE interface with the following details:

- Project Structure:** The left sidebar shows the project structure for "Project8".
  - Root level: intro, settings.py (selected), urls.py, wsgi.py, relationships, user\_auth, migrations, admin.py, apps.py.
  - user\_auth directory:
    - migrations: \_\_init\_\_.py
    - views.py
  - venv library root: db.sqlite3, manage.py.
  - External Libraries and Scratches and Consoles.
- Code Editor (settings.py):** Displays configuration settings.

```
STATIC_URL = '/static/'  
# Default primary key field type  
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field  
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'  
AUTH_USER_MODEL = 'user_auth.User'
```
- Code Editor (models.py):** Displays the User model definition, highlighted with a red box.

```
from django.db import models  
from django.contrib.auth.models import AbstractUser  
  
class User(AbstractUser):  
    is_student = models.BooleanField(default=False)
```
- Bottom Navigation:** TODO, Problems, DB Execution Console, Terminal, Python Console.
- Top Bar:** Navigate, Code, Refactor, Run, Tools, VCS, Window, DB Navigator, Help, pythonProject8 - models.py.

*POWIAZANIE RELACJA JEDEN  
DO JEDNEGO Z MODELEM  
ROZSZERZAJACYM*

# *KIEDY STOSOWAC?*

Kiedy potrzebujemy uzupełnić wbudowany model użytkownika o dodatkowe informacje podawane poza procesem rejestracji (np. profil użytkownika).

# ROZSZERZENIE MODELU *ABSTRACTUSER*

The screenshot shows the PyCharm IDE interface with the following details:

- File Structure:** Shows a tree view of files and folders, including `__init__.py`, `models.py`, `settings.py`, `urls.py`, and `wsgi.py`.
- Code Editor:** The `models.py` file is open, containing the following code:

```
from django.db import models
from django.contrib.auth.models import User
from django.db.models.signals import post_save
from django.dispatch import receiver

class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    bio = models.TextField(max_length=500, blank=True)
    location = models.CharField(max_length=30, blank=True)
    birth_date = models.DateField(null=True, blank=True)

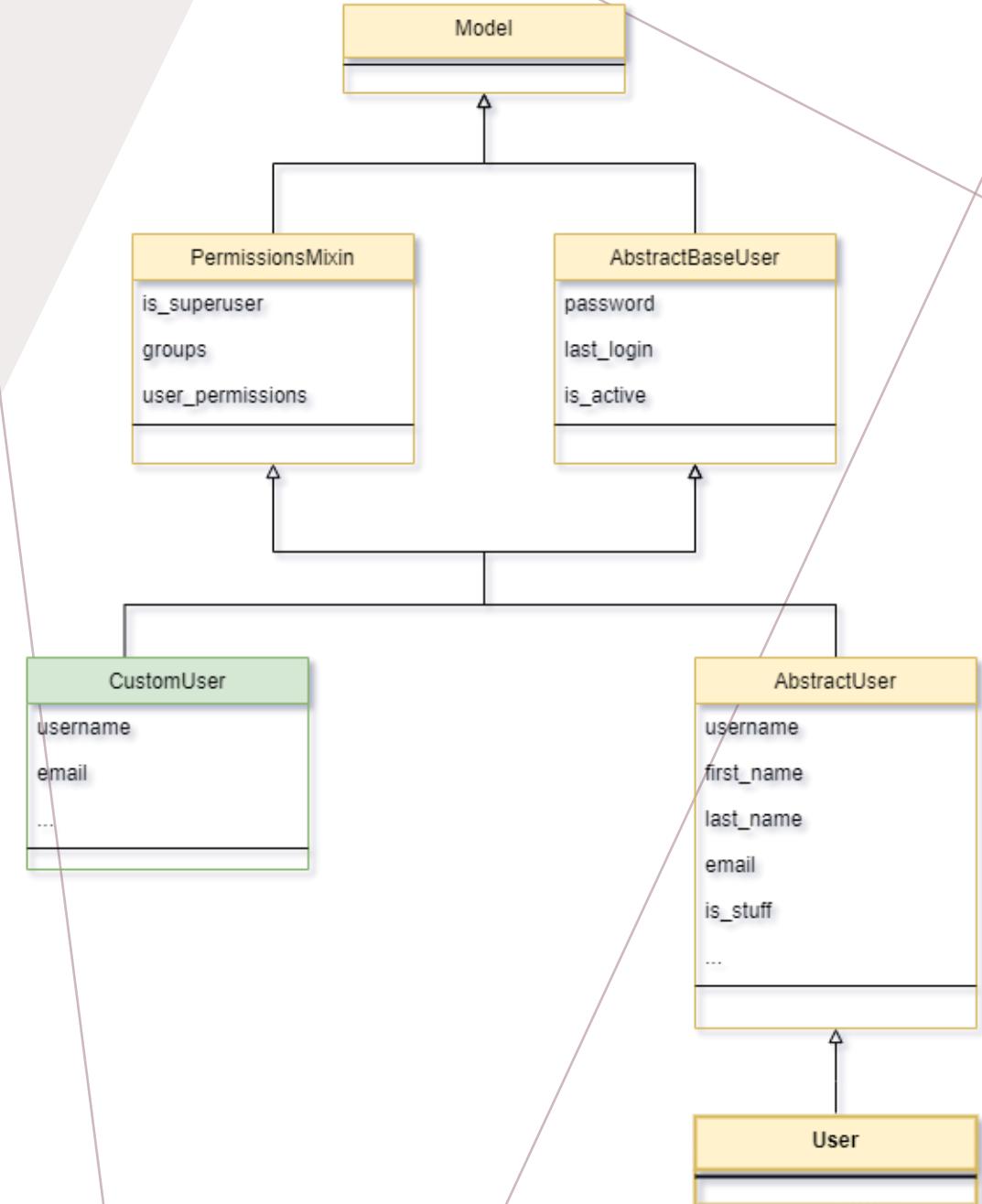
@receiver(post_save, sender=User)
def create_user_profile(sender, instance, created, **kwargs):
    if created:
        Profile.objects.create(user=instance)

@receiver(post_save, sender=User)
def save_user_profile(sender, instance, **kwargs):
    instance.profile.save()
```
- Terminal:** The terminal tab is active, showing the command `(venv) C:\Users\jerem\PycharmProjects\pythonProject8>`.
- Bottom Navigation:** Includes tabs for `TODO`, `Problems`, `DB Execution Console`, `Terminal`, and `Python Console`.

The background of the slide features a light gray rectangular area centered horizontally. This area is surrounded by several thin, dark gray lines that form a complex, abstract geometric pattern. These lines intersect at various points, creating a sense of depth and perspective. Some lines are straight, while others are curved or diagonal, forming a network-like structure.

# *ROZSzerzenie modelu ABSTRACTBASEUSER*

# *ROZSZERZENIE MODELU ABSTRACTBASEUSER*



# *KIEDY STOSOWAC?*

Kiedy chcemy zmienić logikę uwierzytelnienia użytkownika (np. w poświadczaniach zamiast nazwy użytkownika podawać mail).

*WBUDOWANE FORMULARZE  
SYSTEMU  
UWIERZYTELNIANIA*

AdminPasswordChangeForm	PasswordChangeForm	SetPasswordForm	UserCreationForm
AuthenticationForm	PasswordResetForm	UserChangeForm	

*WBUDOWANE FORMULARZE*  
*(MODULU DJANGO.CONTRIB.AUTH.FORMS)*

← → C

127.0.0.1:8000/admin-password/

Password: 

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password (again):  Enter the same password as before, for verification.Zarejestruj się*ADMINPASSWORDCHANGEFORM*

127.0.0.1:8000/login/

Username: Password: [Zarejestruj się](#)*AUTHENTICATIONFORM*

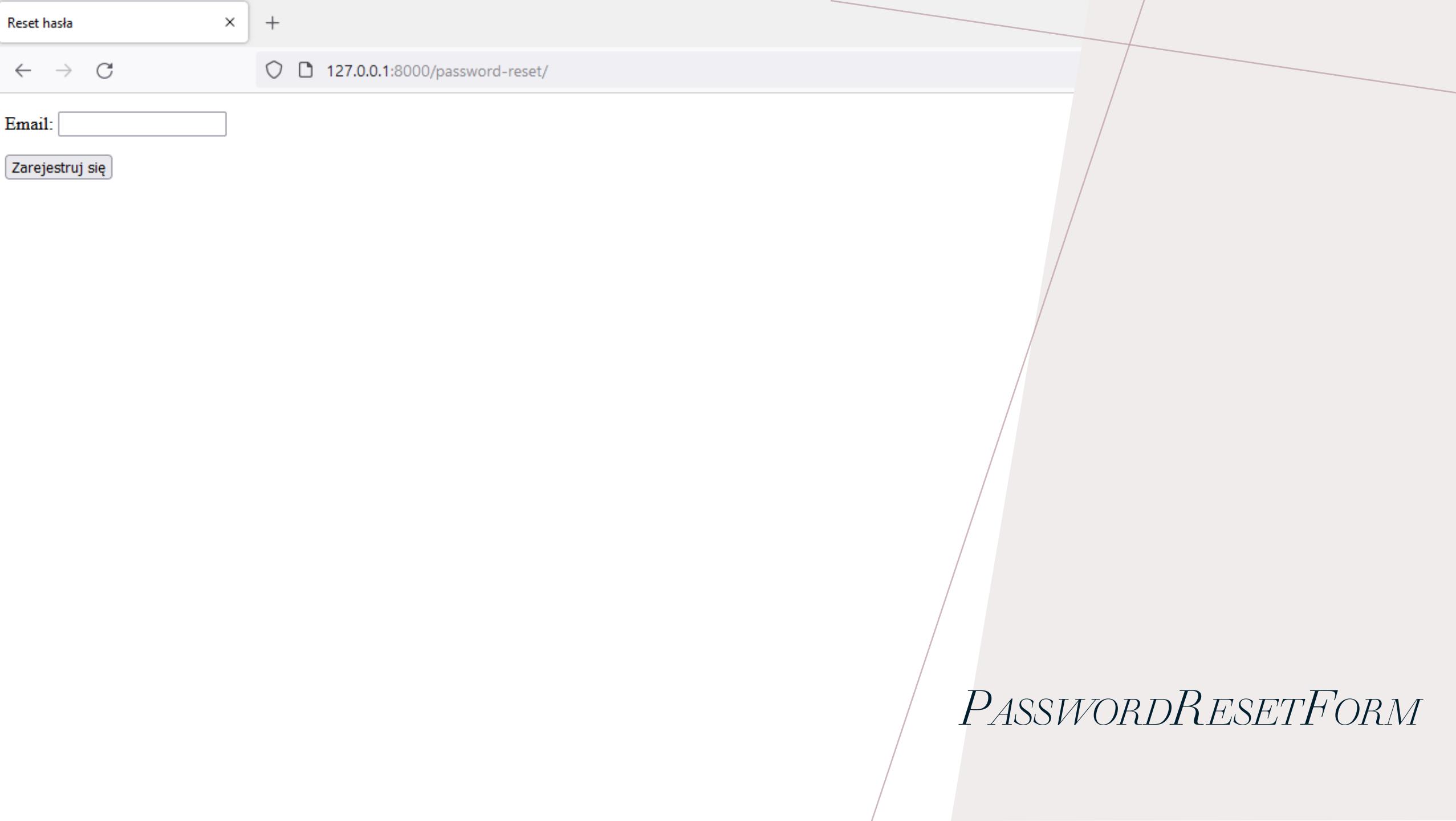
[←](#) [→](#) [⟳](#)

127.0.0.1:8000/password/

Old password: New password: 

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

New password confirmation: [Zarejestruj się](#)*PASSWORDCHANGEFORM*



*PASSWORDRESETFORM*

← → ⟳

127.0.0.1:8000/password-set/

New password: 

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

New password confirmation: Zarejestruj się*SETPASSWORDFORM*

[←](#) [→](#) [↻](#)

127.0.0.1:8000/user/

Password:

**No password set.**Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using [this form](#).Last login: Superuser status:  Designates that this user has all permissions without explicitly assigning them.Groups:  The groups this user belongs to. A user will get all permissions granted to each of their groups.

admin   log entry   Can add log entry
admin   log entry   Can change log entry
admin   log entry   Can delete log entry
admin   log entry   Can view log entry

User permissions:  Specific permissions for this user.Username:  Required. 150 characters or fewer. Letters, digits and @/./+/-/\_ only.First name: Last name: Email address: Staff status:  Designates whether the user can log into this admin site.Active:  Designates whether this user should be treated as active. Unselect this instead of deleting accounts.Date joined: [Zarejestruj się](#)*USERCHANGEFORM*



Username:  Required. 150 characters or fewer. Letters, digits and @/./+/-/\_ only.

Password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation:  Enter the same password as before, for verification.

*USER CREATION FORM*



# *WBUDOWANE ENDPOINTY SYSTEMU UWIERZYTELNIANIA*

login/	password_change/	password_reset/	reset/<uidb64>/<token>/
logout/	password_change/done/	password_reset/done/	reset/done/

# WBUDOWANE ENDPOINTY

*(MODULU DJANGO.CONTRIB.AUTH.URLS)*

auth\_proj &gt; config &gt; urls.py

Project

auth\_proj C:\Users\jerem\PycharmProjects\auth\_proj

- > accounts
- > config
  - \_\_init\_\_.py
  - asgi.py
  - settings.py
  - urls.py
  - wsgi.py
- > newapp
- > venv library root
  - db.sqlite3
  - manage.py
- > External Libraries
- Scratches and Consoles

urls.py x tests.py x models.py x admin.py x

```
10     1. Add an import: from other_app.views import Home
11     2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12 Including another URLconf
13     1. Import the include() function: from django.urls import include, path
14     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15     """
16     from django.contrib import admin
17     from django.urls import path, include
18
19     urlpatterns = [
20         path('admin/', admin.site.urls),
21         path('accounts/', include('django.contrib.auth.urls')),
22     ]
23
24
```

Terminal: Local +

(venv) C:\Users\jerem\PycharmProjects\auth\_proj&gt;python manage.py runserver

Watching for file changes with StatReloader

Performing system checks...

System check identified no issues (0 silenced).

April 14, 2022 - 12:27:25

Django version 4.0.3, using settings 'config.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CTRL-BREAK.

URL APLIKACJI AUTH

auth\_proj &gt; config &gt; urls.py

Project  
DB Browser  
Scratches and Consoles

- auth\_proj C:\Users\jerem\PycharmProjects\auth\_proj
  - accounts
  - config
    - \_\_init\_\_.py
    - asgi.py
    - settings.py
    - urls.py
    - wsgi.py
  - newapp
  - venv library root
    - db.sqlite3
    - manage.py
- External Libraries

```
urls.py x tests.py x models.py x admin.py x
10 1. Add an import: from other_app.views
11 2. Add a URL to urlpatterns: path('' ← → C
12 Including another URLconf
13 1. Import the include() function: f
14 2. Add a URL to urlpatterns: path(
15 """
16     from django.contrib import admin
17     from django.urls import path, include
18
19     urlpatterns = [
20         path('admin/', admin.site.urls)
21         path('accounts/', include('
22     ]
23
24
```

127.0.0.1:8000/accounts/

## Page not found (404)

Request Method: GET

Request URL: http://127.0.0.1:8000/accounts/

Using the URLconf defined in config.urls, Django tried these URL patterns, in this order:

1. admin/
2. accounts/ login/ [name='login']
3. accounts/ logout/ [name='logout']
4. accounts/ password\_change/ [name='password\_change']
5. accounts/ password\_change/done/ [name='password\_change\_done']
6. accounts/ password\_reset/ [name='password\_reset']
7. accounts/ password\_reset/done/ [name='password\_reset\_done']
8. accounts/ reset/<uidb64>/<token>/ [name='password\_reset\_confirm']
9. accounts/ reset/done/ [name='password\_reset\_complete']

The current path, accounts/, didn't match any of these.

You're seeing this error because you have DEBUG = True in your Django settings file. Change this to False if you're seeing this in your production environment.

(venv) C:\Users\jerem\PycharmProjects\auth\_proj&gt;python manage.py

Watching for file changes with StatReloader

Performing system checks...

System check identified no issues (0 silenced).

April 14, 2022 - 12:27:25

Django version 4.0.3, using settings 'config.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CTRL-BREAK.

*URL APLIKACJI AUTH*

*WBUDOWANE WIDOKI  
SYSTEMU UWIERZYTELNIANIA*

LoginView	PasswordChangeView	PasswordResetView	PasswordResetConfirmView
LogoutView	PasswordChangeDoneView	PasswordResetDoneView	PasswordResetCompleteView

# WBUDOWANE WIDOKI

*(MODULU `DJANGO.CONTRIB.AUTH.VIEWS`)*

Project Commit Pull Requests DB Browser Structure Bookmarks

Project

urls.py

```
8
9     urlpatterns = [
10         path("login/", views.LoginView.as_view(), name="login"),
11         path("logout/", views.LogoutView.as_view(), name="logout"),
12         path(
13             "password_change/", views.PasswordChangeView.as_view(), name="password_change"
14         ),
15         path(
16             "password_change/done/",
17             views.PasswordChangeDoneView.as_view(),
18             name="password_change_done",
19         ),
20         path("password_reset/", views.PasswordResetView.as_view(), name="password_reset"),
21         path(
22             "password_reset/done/",
23             views.PasswordResetDoneView.as_view(),
24             name="password_reset_done",
25         ),
26         path(
27             "reset/<uidb64>/<token>/",
28             views.PasswordResetConfirmView.as_view(),
29             name="password_reset_confirm",
30         ),
31         path(
32             "reset/done/",
33             views.PasswordResetCompleteView.as_view(),
34             name="password_reset_complete",
35         ),
36     ],
```

Terminal: Local

Microsoft Windows [Version 10.0.19044.1586]  
(venv) C:\Users\jerem\PycharmProjects\ZDPYTp046\_backend>

WBUDOWANE WIDOKI

*UDZIELANIE DOSTĘPU TYLKO  
UWIERZYTELNIONEMU  
UZYTKOWNIKOWI*

# OPIS

Istnieją trzy popularne sposoby na udzielenie dostępu wyłącznie uwierzytelnionym użytkownikom:

1. Atrybut `is_authenticated` modelu użytkownika
2. Dekorator `login_required` modułu `django.contrib.auth.decorators`
3. Domieszka `LoginRequiredMixin` modułu `django.contrib.auth.mixins`



# *1. ATRYBUT IS AUTHENTICATED*

auth\_proj &gt; newapp &gt; views.py

Project

auth\_proj C:\Users\jerem\PycharmProjects\auth\_proj

- > accounts
- > config
- > newapp
  - > migrations
  - > templates
  - \_\_init\_\_.py
  - admin.py
  - apps.py
  - models.py
  - tests.py
- views.py

venv library root

- db.sqlite3
- manage.py

External Libraries

Scratches and Consoles

views.py

```
from django.shortcuts import render, redirect

def hello_view(request):
    if not request.user.is_authenticated():
        return redirect('login-view')

    return render(
        request,
        'hello.html'
)
```

Terminal: Local (2) × Local × + ▾

Watching for file changes with StatReloader

Performing system checks...

System check identified no issues (0 silenced).

April 14, 2022 - 15:15:22

Django version 4.0.3, using settings 'config.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CTRL-BREAK.

ATRYBUTIS\_AUTHENTICATED

File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help auth\_proj - views.py

\_proj > newapp > views.py

```
File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help auth_proj - views.py
```

```
_proj > newapp > views.py
```

```
settings.py
```

```
118 # https://docs.djangoproject.com/en/4.0/ref/settings/#STATIC_URL
119
120 STATIC_URL = 'static/'
121
122 # Default primary key field type
123 # https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field
124
125 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
126
127 AUTH_USER_MODEL = 'accounts.User'
128
129 LOGIN_URL = 'login-view'
```

```
views.py
```

```
from django.shortcuts import render, redirect
from django.conf import settings

def hello_view(request):
    if not request.user.is_authenticated():
        return redirect(settings.LOGIN_URL)

    return render(
        request,
        'hello.html'
)
```

*ATRYBUT IS\_AUTHENTICATED*

*ZMIENNA LOGIN\_URL*

File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help auth\_proj - views.py

proj > newapp > views.py

views.py

```
from django.shortcuts import render, redirect
from django.conf import settings

def hello_view(request):
    if not request.user.is_authenticated():
        return redirect(f'{settings.LOGIN_URL}?next={request.path}')

    return render(
        request,
        'hello.html'
)
```

A 15 X 5

auth\_proj C:\Users\jeremy\...  
accounts  
config  
\_\_init\_\_.py  
asgi.py  
settings.py  
urls.py  
wsgi.py  
newapp  
migrations  
templates  
\_\_init\_\_.py  
admin.py  
apps.py  
models.py  
tests.py  
views.py  
venv library root  
db.sqlite3  
manage.py  
External Libraries  
Scratches and Consoles

atching for file changes with StatReloader  
erforming system checks...

*ATRYBUT IS\_AUTHENTICATED*

*PRAWIDŁOWA OBSŁUGA PRZEKIEROWAN*

## *2. DEKORATOR LOGIN\_REQUIRED*

auth\_proj &gt; newapp &gt; views.py

The screenshot shows the PyCharm IDE interface. The code editor displays a Python file named 'views.py' with the following content:

```
from django.shortcuts import render
from django.contrib.auth.decorators import login_required

@login_required
def hello_view(request):
    return render(
        request,
        'hello.html'
)
```

The line 'from django.contrib.auth.decorators import login\_required' is underlined in red, indicating a syntax error or warning. The code editor has a light gray background with syntax highlighting for Python keywords and comments.

Terminal: Local (2) × Local × + ▾

Watching for file changes with StatReloader

Performing system checks...

System check identified no issues (0 silenced).

April 14, 2022 - 15:18:58

Django version 4.0.3, using settings 'config.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CTRL-BREAK.

*LOGIN\_REQUIRED*

auth\_proj &gt; newapp &gt; views.py

Project

- auth\_proj C:\Users\jerem\PycharmProjects\auth\_p
  - > accounts
  - > config
  - > newapp
    - > migrations
    - > templates
    - \_\_init\_\_.py
    - admin.py
    - apps.py
    - models.py
    - tests.py
  - views.py
- > venv library root
  - db.sqlite3
  - manage.py
- > External Libraries
- Scratches and Consoles

views.py

```
1  from django.urls import reverse_lazy
2  from django.shortcuts import render
3  from django.contrib.auth.decorators import login_required
4
5
6  @login_required(login_url=reverse_lazy('my-page'))
7  def hello_view(request):
8      return render(
9          request,
10         'hello.html'
11     )
12
13
14
15
16
```

Terminal: Local (2) × Local × + ▾

Watching for file changes with StatReloader

Performing system checks...

System check identified no issues (0 silenced).

April 14, 2022 - 15:21:14

Django version 4.0.3, using settings 'config.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CTRL-BREAK.

*LOGIN\_REQUIRED**PARAMETR LOGIN\_URL*

### *3. DOMIESZKA LOGINREQUIREDMIXIN*

auth\_proj &gt; newapp &gt; views.py

Project

DB Browser

Structure

Bookmarks

Project

- auth\_proj C:\Users\jerem\PycharmProjects\auth\_p
  - accounts
  - config
  - newapp
    - migrations
    - templates
    - \_\_init\_\_.py
    - admin.py
    - apps.py
    - models.py
    - tests.py
  - views.py
- venv library root
  - db.sqlite3
  - manage.py
- External Libraries
- Scratches and Consoles

views.py

```
1 from django.shortcuts import render
2 from django.views import View
3 from django.contrib.auth.mixins import LoginRequiredMixin
4
5
6 class HelloView(LoginRequiredMixin, View):
7     def get(self, request):
8         return render(
9             request,
10            'hello.html'
11        )
12
13
14
15
16
```

Terminal: Local (2) × Local × + ▾

Watching for file changes with StatReloader

Performing system checks...

System check identified no issues (0 silenced).

April 14, 2022 - 15:28:19

Django version 4.0.3, using settings 'config.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CTRL-BREAK.

*LOGINREQUIREDMIXIN*

auth\_proj &gt; newapp &gt; views.py

Project

- auth\_proj C:\Users\jerem\PycharmProjects\auth\_p
  - > accounts
  - > config
  - > newapp
    - > migrations
    - > templates
    - \_\_init\_\_.py
    - admin.py
    - apps.py
    - models.py
    - tests.py
- views.py

venv library root

- db.sqlite3
- manage.py

External Libraries

Scratches and Consoles

views.py x

```
1 from django.urls import reverse_lazy
2 from django.shortcuts import render
3 from django.views import View
4 from django.contrib.auth.mixins import LoginRequiredMixin
5
6
7 class HelloView(LoginRequiredMixin, View):
8     login_url = reverse_lazy('my-page')
9
10    def get(self, request):
11        return render(
12            request,
13            'hello.html'
14        )
15
16
```

Terminal: Local (2) × Local × + ▾

Watching for file changes with StatReloader

Performing system checks...

System check identified no issues (0 silenced).

April 14, 2022 - 15:29:05

Django version 4.0.3, using settings 'config.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CTRL-BREAK.

*LOGINREQUIREDMIXIN**ATRYBUT LOGIN\_URL*

# USER VS ADMIN

## Atrybut is\_staff

Do panelu administratora mogą zalogować się wyłącznie użytkownicy z wartością parametru `is_staff` ustawioną na `True`.

Jeżeli w Twoim projekcie wyróżnić można tylko zwykłego użytkownika oraz administratora, użyj wbudowanego atrybutu **`is_staff`** do rozróżniania typów użytkownika. Dla stron (lub fragmentów stron), których nie chcesz udostępniać zwykłemu użytkownikowi przeciąż odpowiedzialność tego atrybutu. Wystarczy na podstawie wartości atrybutu `is_staff` udostępniać (lub nie) poszczególne widoki (lub fragmenty szablonów).

## Atrybut is\_superuser

Wbudowany model użytkownika posiada drugi, podobny atrybut – `is_superuser`. Użytkownik, który ma ustawioną wartość parametru `is_superuser` na `True` posiada wszystkie istniejące w aplikacji uprawnienia (patrz dalej). Z użyciem tego atrybutu łatwo jest jedną instrukcją nadać użytkownikowi wszystkie dostępły, bez konieczności nadawania kolejno każdego z istniejących dostępów. Wartość `True` parametru `is_superuser` nie pozwala jednak na zalogowanie się użytkownika w panelu administratora (za to odpowiada atrybut `is_staff`).

Administrator to użytkownik, którego atrybuty `is_staff` oraz `is_superuser` mają wartość `True`.

PC File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help auth\_proj - views.py

auth\_proj > newapp > views.py

Project DB Browser

views.py

```
from django.shortcuts import render, redirect
from django.conf import settings

def hello(request):
    if not request.user.is_staff:
        return redirect(f'{settings.LOGIN_URL}?next={request.path}')

    return render(
        request,
        'hello.html'
)
```

Terminal: Local

Performing system checks...

*ATRYBUT IS\_STAFF*

*PRZYKŁAD PRZECIAZENIA ODPOWIEDZIALNOŚCI*

# *AUTORYZACJA*

*FRAMEWORK PERMISSIONS*

# UPRAWNIENIA W DJANGO.CONTRIB.AUTH

Autoryzacja użytkowników w Django jest realizowana za pomocą frameworku django.contrib.auth. Omówienie mechanizmu autoryzacji wymaga wprowadzenia trzech terminów:

1. Role (Roles)
2. Uprawnienia (Permissions)
3. Grupy (Groups)



*ROLE*

# OPIS

Roles (role) **nie są** częścią framework'u django.contrib.auth. Zostały uwzględnione w omówieniu z uwagi na częste, **błędne** powiązywanie ich z uprawnieniami (które są częścią django.contrib.auth).

Logikę przypisującą użytkownikom rolę należy zaimplementować samodzielnie. Istnieje na to kilka sposobów:

- A. Jeżeli użytkownik może posiadać jedną i tylko jedną rolę najprościej dodać nową kolumnę do modelu.
- B. Jeżeli jeden użytkownik może posiadać wiele ról najprościej stworzyć nową tabelę (rola) i połączyć ją z modelem relacją wiele-do-wielu.
- C. Trzecim popularnym (ale najmniej elastycznym) sposobem na obsługę ról w projekcie jest dopisywanie kolumny w modelu użytkownika dla każdej nowo utworzonej roli.

Dopisanie roli do wbudowanego modelu użytkownika jest jego rozszerzeniem, dlatego niezależnie od powyższej klasyfikacji należy uwzględnić również omówione wcześniej, ogólne reguły rozszerzania modelu użytkownika.

File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help auth\_proj - models.py

proj > accounts > models.py

Project auth\_proj C:\Users\jerem\PycharmProject

accounts

migrations

- 0001\_initial.py
- \_init\_.py
- \_\_init\_\_.py
- admin.py
- apps.py
- models.py
- tests.py
- views.py

config

newapp

venv library root

- db.sqlite3
- manage.py

External Libraries

Scratches and Consoles

models.py x

```
1 from django.db import models
2 from django.contrib.auth.models import AbstractUser
3
4 class User(AbstractUser):
5     STUDENT = 1
6     TEACHER = 2
7
8     ROLE_CHOICES = (
9         (STUDENT, 'Student'),
10        (TEACHER, 'Teacher'),
11    )
12
13    role = models.PositiveIntegerField(null=True, blank=True, choices=ROLE_CHOICES)
14
15
16
```

*A. JEDEN UZYTKOWNIK, JEDNA ROLA*

*ROLE NAJPROSCIEJ PRZYPISAC UZYTKOWNIKOM ROZSZERZAJAC MODEL ABSTRACTUSER*

```
0003_clean_role_type.py
0004_remove_user_roles.py
0005_user_role.py
__init__.py
admin.py
apps.py
models.py
tests.py
views.py
```

```
9     ROLE_CHOICES = (
10         (STUDENT, 'Student'),
11         (TEACHER, 'Teacher'),
12     )
13     role = models.PositiveIntegerField(choices=ROLE_CHOICES)
```

```
Terminal: Local + ▾
```

```
>>> from django.contrib.auth import get_user_model
>>> User = get_user_model()
>>> user = User.objects.first()
>>>
>>> user.role = user.STUDENT
>>> user.save()
>>>
>>> user.role == User.TEACHER
False
>>> user.role == User.STUDENT
True
>>>
```

Przypisanie roli

Sprawdzenie roli

*A. JEDEN UZYTKOWNIK, JEDNA ROLA*

*SPOSOB UZYCIA*

File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help auth\_proj - models.py

proj > accounts > models.py

Project main ▾

auth\_proj C:\Users\jerem\PycharmProject

accounts

migrations

- 0001\_initial.py
- 0002\_role\_user\_roles.py
- 0003\_alter\_role\_type.py
- \_\_init\_\_.py

\_\_init\_\_.py

admin.py

apps.py

models.py

tests.py

views.py

config

- \_\_init\_\_.py
- asgi.py
- settings.py
- urls.py
- wsgi.py

venv library root

- db.sqlite3
- manage.py

External Libraries

models.py

admin.py

```
from django.db import models
from django.contrib.auth.models import AbstractUser

class User(AbstractUser):
    roles = models.ManyToManyField('Role')

class Role(models.Model):
    type = models.CharField(max_length=64)
```

terminal: Local +

## B. JEDEN UZYTKOWNIK, WIELE ROL

JEZELI UZYTKOWNIK MOZE POSIADAC WIECEJ NIZ JEDNA ROLE NAJPROSCIEJ STWORZYC  
DLA ROLI NOWA TABELA I POLACZYC JĄ Z TABELA USER RELACJA WIELE-DO-WIELU

The screenshot shows the PyCharm IDE interface. On the left, there's a file tree with files like \_\_init\_\_.py, admin.py, apps.py, models.py, tests.py, views.py, config, and \_\_init\_\_.py. The main editor window displays Python code for Django models:

```
5 class User(AbstractUser):
6     roles = models.ManyToManyField('Role')
7
8
9 class Role(models.Model):
```

```
Terminal: Local + ▾
>>> from django.contrib.auth import get_user_model
>>> from accounts.models import Role
>>> User = get_user_model()
>>> user = User.objects.first()
>>>
>>> user.roles.add(Role.objects.get(type='student')) ← Przypisanie roli
>>>
>>> user.roles.filter(type='teacher').exists()
False
>>> user.roles.filter(type='student').exists()
True
>>>
```

Przypisanie roli  
Sprawdzenie roli

*B. JEDEN UZYTKOWNIK, WIELE ROL*

*SPOSOB UZYCIA*

File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help auth\_proj - models.py

.proj accounts > models.py

models.py

```
from django.db import models
from django.contrib.auth.models import AbstractUser

class User(AbstractUser):
    is_student = models.BooleanField(default=False)
    is_teacher = models.BooleanField(default=False)
```

auth\_proj C:\Users\jerem\Py...  
accounts  
migrations  
\_\_init\_\_.py  
admin.py  
apps.py  
models.py  
tests.py  
views.py  
config  
newapp  
migrations  
\_\_init\_\_.py  
admin.py  
apps.py  
models.py  
tests.py  
views.py  
venv library root  
db.sqlite3  
manage.py  
External Libraries  
Scratches and Consoles

Terminal: Local +

*C. ODDZIELNA KOLUMNA DLA KAZDEJ ROLI*

*WYGODNE W PRZYPADKU NIEDUZEJ, STAŁEJ LICZBY ROLI*

```
DB
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
config
__init__.py
asgi.py

Terminal: Local (2) × Local × + ×

>>> from django.contrib.auth import get_user_model
>>> User = get_user_model()
>>> user = User.objects.first()
>>>
>>> user.is_student = True
>>> user.save()
>>>
>>> user.is_student
True
>>> user.is_teacher
False
>>>
```

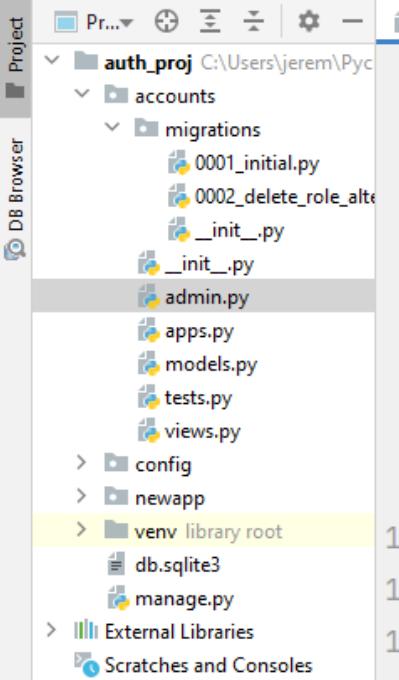
Przypisanie roli

Sprawdzenie roli

*C. ODDZIELNA KOLUMNNA DLA KAZDEJ ROLI*

*SPOSÓB UŻYCIA*

auth\_proj &gt; accounts &gt; admin.py



```
from django.contrib import admin
from django.contrib.auth.admin import UserAdmin
from django.contrib.auth.forms import UserChangeForm
from django.contrib.auth import get_user_model

User = get_user_model()

class CustomUserChangeForm(UserChangeForm):
    class Meta(UserChangeForm.Meta):
        model = User

class CustomUserAdmin(UserAdmin):
    form = CustomUserChangeForm

    fieldsets = UserAdmin.fieldsets + (
        (None, {'fields': ('role',)}),
    )

admin.site.register(User, CustomUserAdmin)
```

*WYSWIETLANIE WLASNEGO  
MODELU UZYTKOWNIKA W  
PANELU ADMINISTRATORA  
(JEDEN UZYTKOWNIK, JEDNA ROLA)*

Quit the server with CTRL-BREAK.

# PODSUMOWANIE

Role to sposób na pogrupowanie użytkowników zgodnie z funkcją jaką pełnią w projekcie. Ról używamy w zapytaniach do bazy danych. Nie używamy ich do zarządzania uprawnieniami. Do zarządzania uprawnieniami wykorzystujemy model Permission lub Group (zarządzanie grupami uprawnień). Grupy mogą pokrywać się z rolami (ale nie muszą).

# *UPRAWNIENIA*

# OPIS

Permissions (uprawnienia) to zbiór reguł pozwalający użytkownikom (lub grupom użytkowników) na wykonanie wybranych operacji. Django posiada wbudowany system uprawnień, który umożliwia przypisywanie i odbieranie uprawnień użytkownikom (lub grupom użytkowników).

Wbudowana aplikacja `django.contrib.auth` automatycznie tworzy cztery uprawnienia CRUD (add, view, change, delete) dla każdego nowo powstałego modelu. Wszystkie uprawnienia są przechowywane w modelu **`django.contrib.auth.models.Permission`**. Model użytkownika jest w relacji wiele-do-wielu (atrybut `user_permissions`) z modelem `Permission` poprzez klasę `PermissionsMixin`.

Automatycznie wygenerowane uprawnienia możemy wykorzystywać w logice naszej aplikacji. Możemy również tworzyć własne uprawnienia. Każde uprawnienie powiązane jest z jakimś modelem, dlatego podczas tworzenia nowego uprawnienia należy zawsze wskazać, którego modelu to uprawnienie będzie dotyczyć (np. poprzez atrybut `content_type`).

auth\_proj > venv > Lib > site-packages > django > contrib > auth > models.py

```
37 class Permission(models.Model):
38
39     name = models.CharField(_("name"), max_length=255)
40     content_type = models.ForeignKey(
41         ContentType,
42         models.CASCADE,
43         verbose_name=_("content type"),
44     )
45     codename = models.CharField(_("codename"), max_length=100)
46
47     objects = PermissionManager()
48
49     class Meta:
50         verbose_name = _("permission")
51         verbose_name_plural = _("permissions")
52         unique_together = [[ "content_type", "codename"] ]
53         ordering = [ "content_type__app_label", "content_type__model", "codename"]
```

DJANGO.CONTRIB.AUTH.MODELS.PERMISSION

# OPERACJA NA MODELU PERMISSION

Dla modelu Permission obowiązują wszystkie poznane przy omawianiu Django ORM zasady. W szczególności CRUD modelu Permission nie różni się niczym od CRUD na innych modelach. Istnieją jednak kilka różnic. Najważniejsze z nich to mechanizm automatycznego tworzenia oraz dodatkowe narzędzia do weryfikacji uprawnień.

# *PERMISSION CRUD*

The screenshot shows a PyCharm IDE interface with the following details:

- File Structure:** Shows files like `apps.py`, `models.py`, `tests.py`, and `views.py`.
- Terminal:** Local (2) tab is active.
- Code:** Python code for creating a custom permission.

```
class Grade(models.Model):
    point = models.PositiveIntegerField(choices=[(item, f'{item}') for item in range(1,7)])
```

```
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from django.contrib.auth.models import Permission
>>> from django.contrib.contenttypes.models import ContentType
>>>
>>> from newapp.models import Grade
>>>
>>> content_type = ContentType.objects.get_for_model(Grade)
>>> permission = Permission.objects.create(
...     codename='can_custom_permission',
...     name='Can Custom Permission',
...     content_type=content_type
... )
>>>
```

A red rectangular box highlights the line of code where the `content_type` variable is assigned.

Aktywuj system Windows  
Przejdz do ustawien, aby aktywować system Windows

# UPRAWNIENIA – TWORZENIE (C)

The screenshot shows a Python development environment with the following code in the editor:

```
models.py
tests.py
views.py
> venv library root
db.sqlite3

Terminal: Local (2) × Local × + ▾

>>>
>>> from newapp.models import Grade
>>>
>>> content_type = ContentType.objects.get_for_model(Grade)
>>> permission = Permission.objects.create(
...     codename='can_custom_permission',
...     name='Can Custom Permission',
...     content_type=content_type
... )
>>>
>>> permission = Permission.objects.get(codename='can_custom_permission')
>>> permission
<Permission: newapp | grade | Can Custom Permission>
>>>
```

A red bracket highlights the line `>>> permission = Permission.objects.get(codename='can_custom_permission')`. The terminal output shows the creation of a permission object named "Can Custom Permission".

Aktywuj system Windows  
Przejdz do ustawieñ, aby aktywowaæ system Windows

# UPRAWNIENIA – ODCZYT (R)

```
6
7     class Grade(models.Model):
8         point = models.PositiveIntegerField(choices=[(item, f'{item}') for item in range(1,7)])
```

Terminal: Local (2) × Local × + ▾

```
...     codename='can_custom_permission',
...     name='Can Custom Permission',
...     content_type=content_type
... )
>>>
>>> permission = Permission.objects.get(codename='can_custom_permission')
>>> permission
<Permission: newapp | grade | Can Custom Permission>
>>>
>>> permission.name = "Can custom permission"
>>> permission.save()
>>> permission
<Permission: newapp | grade | Can custom permission>
>>>
```

Aktywuj system Windows  
Przejdz do ustawieñ, aby aktywowaæ system Windows

Version Control TODO Problems Python Packages Python Console Terminal

1:1 CRLF UTF-8 4 spaces Python 3.8 (auth)

# UPRAWNIENIA – MODYFIKACJA (U)

```
7     class Grade(models.Model):
8         point = models.PositiveIntegerField(choices=[(item, f'{item}') for item in range(1,7)])
...
... )
>>>
>>> permission = Permission.objects.get(codename='can_custom_permission')
>>> permission
<Permission: newapp | grade | Can Custom Permission>
>>>
>>> permission.name = "Can custom permission"
>>> permission.save()
>>> permission
<Permission: newapp | grade | Can custom permission>
>>>
>>> permission.delete()
(1, {'auth.Permission': 1})
>>>
```

Aktywuj system Windows  
Przejdz do ustawieñ, aby aktywowaæ system Windows

# UPRAWNIENIA – USUWANIE (D)



# *PRZYPISYWANIE UPRAWNIEN DO UZYTKOWNIKA*

# UPRAWNIENIA UZYTKOWNIKA

Tabela uprawnienia połączona jest z tabelą użytkownicy relacją wiele do wielu. Relacje reprezentuje atrybut `user_permissions` klasy `PermissionsMixin`.

Project DB Browser Structure Bookmarks

models.py

```
203     class PermissionsMixin(models.Model):
204         """
205             Add the fields and methods necessary to support the Group and Permission
206             models using the ModelBackend.
207         """
208
209         is_superuser = models.BooleanField(
210             _("superuser status"),
211             default=False,
212             help_text=_(
213                 "Designates that this user has all permissions without "
214                 "explicitly assigning them."
215             ),
216         )
217         groups = models.ManyToManyField(
218             Group,
219             verbose_name=_("groups"),
220             blank=True,
221             help_text=_(
222                 "The groups this user belongs to. A user will get all permissions granted to each of their groups."
223             ),
224             related_name="user_set",
225             related_query_name="user",
226         )
227         user_permissions = models.ManyToManyField(
228             Permission,
229             verbose_name=_("user permissions"),
230             blank=True,
231             help_text=_("Specific permissions for this user."),
232             related_name="user_set",
233             related_query_name="user",
234         )
```

PermissionsMixin

Terminal: Local (2) × Local × Local (3) × + ▾

```
>>> view_grade_perm.user_set.add(student)
>>> [ ]
```

Version Control TODO Problems Python Packages Python Console Terminal

*UPRAWNIENIA  
UZYTKOWNIKA*

```
7     class Grade(models.Model):
8         point = models.PositiveIntegerField(choices=[(item, f'{item}') for item in range(1,7)])
```

Terminal: Local (2) × Local × Local (3) × + ▾

Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

(InteractiveConsole)

>>>

```
>>> from django.contrib.auth import get_user_model
>>> from django.shortcuts import get_object_or_404
>>> from django.contrib.auth.models import Permission
>>>
>>> User = get_user_model()
>>> student = get_object_or_404(User, username='Sam')
>>> view_grade_perm = get_object_or_404(Permission, codename='view_grade')
>>>
>>> student.user_permissions.add(view_grade_perm)
>>> █
```

Aktywuj system Windows  
Przejdz do ustawień, aby aktywować system Windows

Version Control TODO Problems Python Packages Python Console Terminal

1:1 CRLF UTF-8 4 spaces Python 3.8 (auth\_

# PRZYPISANIE UPRAWNIEN DO UZYTKOWNIKA

(OD STRONY UZYTKOWNIKA)

```
6  
7     class Grade(models.Model):  
8         point = models.PositiveIntegerField(choices=[(item, f'{item}') for item in range(1,7)])
```

Terminal: Local (2) × Local × Local (3) × + ▾

Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

(InteractiveConsole)

>>>

```
>>> from django.contrib.auth import get_user_model  
>>> from django.shortcuts import get_object_or_404  
>>> from django.contrib.auth.models import Permission  
>>>  
>>> User = get_user_model()  
>>> student = get_object_or_404(User, username='Sam')  
>>> view_grade_perm = get_object_or_404(Permission, codename='view_grade')  
>>>  
>>> view_grade_perm.user_set.add(student)  
>>> █
```

Aktywuj system Windows  
Przejdz do ustawieñ, aby aktywowaæ system Windows

Version Control TODO Problems Python Packages Python Console Terminal

1:1 CRLF UTF-8 4 spaces Python 3.8 (auth)

# PRZYPISANIE UPRAWNIEN DO UZYTKOWNIKA

(OD STRONY UPRAWNIEN)

# CACHOWANIE UPRAWNIEN

Uwaga!

Uprawnienia są cachowane. Dlatego po nadaniu uprawnienia użytkownikowi należy przed weryfikacją czy użytkownik istotnie posiada już nadane mu uprawnienie wczytać ponownie z bazy dane dotyczące użytkownika.

# *MECHANIZM AUTOMATYCZNEGO TWORZENIA*

*(MODEL LEVEL PERMISSION)*

## MODEL LEVEL PERMISSION

Stworzenie nowego modelu automatycznie generuje 4 nowe wpisy w tabeli z uprawnieniami. Automatycznie wygenerowane uprawnienia mają wartość atrybutu codename odpowiednio:

- add\_<model\_name>,
- view\_<model\_name>,
- change\_<model\_name>,
- delete\_<model\_name>.

The screenshot shows the PyCharm IDE interface with the following details:

- Top Bar:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, DB Navigator, Help.
- Project Bar:** auth\_proj > newapp > models.py
- Project Tree:** auth\_proj (C:\Users\jerem\PycharmProject) contains accounts (migrations: 0001\_initial.py, \_\_init\_\_.py, \_\_init\_\_.py, admin.py, apps.py, models.py, tests.py, views.py), config, newapp (migrations: \_\_init\_\_.py, admin.py, apps.py, models.py, tests.py, views.py), venv (library root: db.sqlite3, manage.py), External Libraries, and Scratches and Consoles.
- Code Editor:** The models.py file is open, showing Python code for a Grade model. The code includes imports for django.db.models and django.contrib.auth.get\_user\_model, defines a User variable, and creates a Grade class with fields: point (PositiveIntegerField), student (OneToOneField to User), teacher (OneToOneField to User), added (DateTimeField with auto\_now=True), and modified (DateTimeField with auto\_now=True).
- Terminal:** Local terminal window showing the command: (venv) C:\Users\jerem\PycharmProjects\auth\_proj>
- Bottom Navigation:** Version Control, TODO, Problems, Python Packages, Python Console, Terminal.

Stworzenie tabeli Grade wygeneruje w tabeli Permission cztery nowe wpisy o wartości w kolumnie codename odpowiednio:

- add\_grade
- view\_grade
- change\_grade
- delete\_grade

admin | Change user | Django sit x +

127.0.0.1:8000/admin/accounts/user/1/change/

Superuser status  
Designates that this user has all permissions without explicitly assigning them.

Groups:



The groups this user belongs to. A user will get all permissions granted to each of their groups. Hold down "Control", or "Command" on a Mac, to select more than one.

User permissions:

- [contenttypes | content type | Can change content type](#)
- [contenttypes | content type | Can delete content type](#)
- [contenttypes | content type | Can view content type](#)
- [newapp | grade | Can add grade](#)
- [newapp | grade | Can change grade](#)
- [newapp | grade | Can delete grade](#)
- [newapp | grade | Can view grade](#)
- [sessions | session | Can add session](#)
- [sessions | session | Can change session](#)

Specific permissions for this user. Hold down "Control", or "Command" on a Mac, to select more than one.

## DOMYSLNA REPREZENTACJA NAPISOWA MODELU PERMISSION

<NAZWA APLIKACJI> | <NAZWA MODELU> | <NAZWA UPRAWNIENIA>

Email address: admin@admin.com

Staff status

Designates whether the user can log into this admin site.

Active

Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

Date joined:

Datum: 2023-04-11

*WIDOK UPRAWNIEN W  
PANELU ADMINISTRATORA*

*(REPREZENTACJA NAPISOWA)*

VCS Window DB Navigator Help auth\_proj - models.py

```
py <py>
from django.db import models
from django.contrib.auth import get_user_model

User = get_user_model()

class Grade(models.Model):
    point = models.PositiveIntegerField(choices=[(item, f'{item}') for item in range(1,7)])
    student = models.OneToOneField(User, on_delete=models.CASCADE, related_name='student')
    teacher = models.OneToOneField(User, on_delete=models.CASCADE, related_name='teacher')
    added = models.DateTimeField(auto_now_add=True)
    modified = models.DateTimeField(auto_now=True)

    class Meta:
        permissions = (
            ('first_custom_permission', 'can first custom permission'),
            ('second_custom_permission', 'can second custom permission')
        )

zelkie prawa zastrzezone.

Projects\auth_proj>
```

Aktywuj system Windows  
Przejdz do ustawień, aby aktywować

26:1 CRLF UTF-8

Za pomocą atrybutu **permissions** klasy **Meta** do czterech automatycznie wygenerowanych dla modelu uprawnień możemy dodać dodatkowe, własne uprawnienia, zgodnie ze schematem:

```
class Meta:
    permissions = (
        <codename>, <name>
    )
```

Po wygenerowaniu i zastosowaniu migracji odpowiednie wpisy zostaną umieszczone w tabeli permission.

# *MECHANIZM WERYFIKACJI*

# KONWENCJA NAZEWNICZA

Przy weryfikacji uprawnień odwołujemy się do poszczególnych uprawnień zgodnie z konwencją:

*'<app\_name>.<codename>'*

, gdzie codename to atrybut modelu Permission

# KONWENCJA NAZEWNICZA

Do 4 automatycznie wygenerowanych uprawnień odwołujemy się za pomocą nazw:

- '*<app\_name>.add\_<model\_name>*'
- '*<app\_name>.view\_<model\_name>*'
- '*<app\_name>.change\_<model\_name>*'
- '*<app\_name>.delete\_<model\_name>*'

The screenshot shows the PyCharm IDE interface with the following details:

- File Menu:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, DB Navigator, Help.
- Project Bar:** auth\_proj > newapp > models.py
- Project Tree:** auth\_proj (C:\Users\jerem\PycharmProjects) contains accounts (migrations: 0001\_initial.py, \_\_init\_\_.py, \_\_init\_\_.py, admin.py, apps.py, models.py, tests.py, views.py), config, newapp (migrations: \_\_init\_\_.py, admin.py, apps.py, models.py, tests.py, views.py), venv (library root: db.sqlite3, manage.py), External Libraries, and Scratches and Consoles.
- Code Editor:** The models.py file is open, showing Python code for a Grade model. The code includes imports for django.db.models and django.contrib.auth.get\_user\_model, defines a User variable, and creates a Grade class with fields point, student, teacher, added, and modified.
- Terminal:** The terminal shows the command (venv) C:\Users\jerem\PycharmProjects\auth\_proj>.
- Bottom Navigation:** Version Control, TODO, Problems, Python Packages, Python Console, Terminal.

Do cztery automatycznie wygenerowane uprawnienia dla modelu Grade odwołujemy się zgodnie z konwencją:

- 'newapp.add\_grade'
- 'newapp.view\_grade'
- 'newapp.change\_grade'
- 'newapp.delete\_grade'

# WERYFIKACJA UPRAWNIEN

Django posiada 4 sposoby weryfikacji uprawnień:

1. metoda *has\_perm* modelu użytkownika (w kodzie)
2. dekorator *permission\_required*(dla widoków funkcyjnych)
3. domieszka *PermissionRequiredMixin*(dla widoków klasowych)
4. zmienna kontekstowa *perms*(w szablonach)

# *1. METODA HAS\_PERM*

auth\_proj &gt; newapp &gt; views.py

Project

auth\_proj C:\Users\jerem\PycharmProjects  
  accounts  
    migrations  
    templates  
      \_\_init\_\_.py  
      admin.py  
      apps.py  
      models.py  
      tests.py  
      urls.py  
      views.py  
  config  
    \_\_init\_\_.py

views.py x

```
1  from django.shortcuts import render, redirect
2  from django.conf import settings
3  from django.contrib.auth.models import User
4
5
6  def hello(request):
7      if not request.user.is_staff:
8          return redirect(f'{settings.LOGIN_URL}?next={request.path}'
```

Terminal: Local x Local (2) x + ▾

Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

(InteractiveConsole)

```
>>> from django.contrib.auth import get_user_model
>>> from django.shortcuts import get_object_or_404
>>>
>>> User = get_user_model()
>>> student = get_object_or_404(User, username="Sam")
>>>
>>> student.has_perm('newapp.view_grade')
True
>>> student.has_perm('newapp.add_grade')
False
>>> 
```

## UZYCIE UPRAWNIEN

W KODZIE  
(METODA HAS\_PERM)

## *2. DEKORATOR PERMISSION\_REQUIRED*

## 2. DEKORATOR PERMISSION\_REQUIRED

Zamiast samodzielnie implementować obsługę uprawnień w poszczególnych widokach możemy użyć dekoratora **permission\_required** z modułu django.contrib.auth.decorators. Dekorator jako parametr przyjmuje nazwę weryfikowanego uprawnienia. Domyślnie, w przypadku braku takiego uprawnienia użytkownik zostaje przekierowany na widok logowania (na widok znajdujący się na endpointie wskazywanym przez wartość zmiennej **LOGIN\_URL**, domyślnie jest to 'accounts/login/'). Nie jest to dobre rozwiązanie, ponieważ w ten sposób zalogowany użytkownik zobaczy ekran logowania, a nie informacje o braku uprawnień.

Jeżeli zamiast przekierowywać użytkownika chcemy, żeby użytkownik otrzymywał kod 403 (Forbidden Error) należy użyć opcjonalnego parametru **raised\_exception**. Ustawienie wartości parametru `raised_exception=True` spowoduje, że użytkownikowi bez uprawnień będzie wyświetlał się błąd 403.

Jeżeli natomiast chcemy, żeby użytkownik był przekierowywany na widok inny niż ten wskazywany przez zmienną **LOGIN\_URL**, należy użyć opcjonalnego parametru **login\_url**.

auth\_proj &gt; newapp &gt; views.py

Project

DB Browser

Structure

Bookmarks

views.py

```
1 from django.shortcuts import HttpResponseRedirect  
2 from django.contrib.auth.decorators import permission_required  
3  
4  
5 @permission_required('newapp.change_grade')  
6 def update_grade_view(request):  
7     return HttpResponseRedirect("Hello!")
```

```
venv library root  
db.sqlite3  
manage.py  
External Libraries  
Scratches and Consoles
```

Terminal: Local +

Performing system checks...

System check identified no issues (0 silenced).

April 12, 2022 - 10:17:55

Django version 4.0.3, using settings 'config.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CTRL-BREAK.

*PERMISSION\_REQUIRED**PODSTAWOWE UZYCIE*

auth\_proj &gt; newapp &gt; views.py

Project

DB Browser

Structure

Bookmarks

views.py

```
1 from django.shortcuts import HttpResponseRedirect  
2 from django.contrib.auth.decorators import permission_required  
3  
4  
5 @permission_required('newapp.change_grade', raise_exception=True)  
6 def update_grade_view(request):  
7     return HttpResponseRedirect("Hello!")  
8  
9  
10  
11  
12  
13  
14  
15  
16
```

Terminal: Local

Performing system checks...

System check identified no issues (0 silenced).

April 12, 2022 - 10:27:47

Django version 4.0.3, using settings 'config.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CTRL-BREAK.

*PERMISSION\_REQUIRED**PARAMETR RAISE\_EXCEPTION*

The screenshot shows a PyCharm IDE interface with the following details:

- Project Tree:** Shows the project structure with files like models.py, tests.py, urls.py, views.py, config/\_init\_.py, asgi.py, settings.py, urls.py, and wsgi.py.
- Code Editor:** The file views.py contains the following Python code:

```
from django.shortcuts import HttpResponseRedirect
from django.contrib.auth.decorators import permission_required

@permission_required('newapp.change_grade', raise_exception=True)
def update_grade_view(request):
    return HttpResponseRedirect("Hello!")
```
- Terminal:** Shows the command line output:

```
Django version 4.0.0,
Starting development server at http://127.0.0.1:8000
Quit the server with CTRL-BREAK.
```
- Browser:** A window titled "403 Forbidden" is displayed, indicating that the user does not have the required permission to access the page.

**Text Labels:**

- PERMISSION\_REQUIRED**: A large, bold, italicized text label positioned on the right side of the image.
- PARAMETR RAISE\_EXCEPTION**: A large, bold, italicized text label positioned below the first one.

auth\_proj &gt; newapp &gt; views.py

Project

DB Browser

Structure

Bookmarks

newapp\views.py accounts\urls.py config\urls.py accounts\views.py

```

1  from django.shortcuts import HttpResponseRedirect, reverse
2  from django.urls import reverse_lazy
3  from django.contrib.auth.decorators import permission_required
4
5
6  @permission_required('newapp.change_grade', login_url=reverse_lazy('my-page'))
7  def update_grade_view(request):
8      return HttpResponseRedirect("Hello!")
9
10
11
12
13
14
15
16
    hi()

```

Terminal: Local +

System check identified no issues (0 silenced).

April 12, 2022 - 12:18:11

Django version 4.0.3, using settings 'config.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CTRL-BREAK.

[12/Apr/2022 12:18:19] "GET /hello/ HTTP/1.1" 302 0

[12/Apr/2022 12:18:19] "GET /hi/?next=/hello/ HTTP/1.1" 200 3

*PERMISSION\_REQUIRED**PARAMETER LOGIN\_URL*



### *3. DOMIESZKA PERMISSIONREQUIREDMIXIN*

## 2. DOMIESZKA PERMISSIONREQUIREDMIXIN

Domieszka **PermissionRequiredMixin** jest tym dla widoków klasowym, czym dekorator `permission_required` dla widoków funkcyjnych. Posiada ten sam zestaw parametrów, a dodatkowo może przyjmować zarówno pojedyncze uprawnienie jak i typ iterowalny zawierający listę uprawnień.

auth\_proj &gt; newapp &gt; views.py

Project

DB Browser

Structure

Bookmarks

views.py

```
1  from django.shortcuts import HttpResponseRedirect
2  from django.views import View
3  from django.contrib.auth.mixins import PermissionRequiredMixin
4
5
6  class UpdateGradeView(PermissionRequiredMixin, View):
7      permission_required = 'newapp.change_grade'
8
9      def get(self):
10          return HttpResponseRedirect("Hello!")
```

Terminal: Local +

Performing system checks...

System check identified no issues (0 silenced).

April 12, 2022 - 12:51:55

Django version 4.0.3, using settings 'config.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CTRL-BREAK.

PERMISSIONREQUIREDMIXIN

N

PODSTAWOWE UZYCIE

auth\_proj &gt; newapp &gt; views.py

```
from django.shortcuts import HttpResponseRedirect
from django.views import View
from django.contrib.auth.mixins import PermissionRequiredMixin

class ChangeGradeView(PermissionRequiredMixin, View):
    permission_required = (
        'newapp.create_grade',
        'newapp.change_grade',
        'newapp.delete_grade',
    )

    def get(self):
        return HttpResponseRedirect("Hello!")

ChangeGradeView
```

Terminal: Local +

Performing system checks...

System check identified no issues (0 silenced).

April 12, 2022 - 12:57:36

Django version 4.0.3, using settings 'config.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CTRL-BREAK.

*PERMISSIONREQUIREDMIXIN*

N

*LISTA UPRAWNIEN*

auth\_proj &gt; newapp &gt; views.py

Project DB Browser

```

from django.shortcuts import HttpResponseRedirect
from django.views import View
from django.contrib.auth.mixins import PermissionRequiredMixin

class UpdateGradeView(PermissionRequiredMixin, View):
    permission_required = 'newapp.change_grade'
    raise_exception = True

    def get(self):
        return HttpResponseRedirect("Hello!")

```

Terminal: Local +

Performing system checks...

System check identified no issues (0 silenced).

April 12, 2022 - 12:59:37

Django version 4.0.3, using settings 'config.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CTRL-BREAK.

*PERMISSIONREQUIREDMIXIN**N**PARAMETER  
RAISED\_EXCEPTION*

The screenshot shows the PyCharm IDE interface. The top navigation bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, DB Navigator, and Help. The current file is 'views.py' under the 'newapp' directory of the 'auth\_proj' project. The code editor displays the following Python code:

```
from django.shortcuts import HttpResponseRedirect
from django.views import View
from django.contrib.auth.mixins import PermissionRequiredMixin

class UpdateGradeView(PermissionRequiredMixin, View):
    permission_required = 'newapp.change_grade'
    raise_exception = True
```

The 'raise\_exception' line is underlined with a red error squiggle. Below the code editor is a browser window showing a 403 Forbidden error page. The bottom status bar indicates the server is running.

*PERMISSIONREQUIREDMIXIN*

*PARAMETR  
RAISED\_EXCEPTION*

auth\_proj &gt; newapp &gt; views.py

Project

DB Browser

```

from django.urls import reverse_lazy
from django.shortcuts import HttpResponseRedirect
from django.views import View
from django.contrib.auth.mixins import PermissionRequiredMixin

class UpdateGradeView(PermissionRequiredMixin, View):
    permission_required = 'newapp.change_grade'
    login_url = reverse_lazy('my-page')

    def get(self):
        return HttpResponseRedirect("Hello!")

```

Terminal: Local +

Performing system checks...

System check identified no issues (0 silenced).

April 12, 2022 - 13:00:18

Django version 4.0.3, using settings 'config.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CTRL-BREAK.

*PERMISSIONREQUIREDMIXIN*

N

*PARAMETR LOGIN\_URL*

## *4. ZMIENNA KONTEKSTOWA PERMS*

## 4. ZMIENNA KONTEKSTOWA PERMS

Zmiennej kontekstowej **perms** używamy w celu ograniczenia dostępu do treści wyświetlanych na stronie użytkownikom nie posiadającym odpowiednich uprawnień. Zmienna jest automatycznie tworzona przez procesor kontekstu szablonów (analogicznie jak na przykład zmienna kontekstowa user).

Zmiennej perms używamy razem ze znacznikiem if zgodnie z konwencją:

*{% if perms.<app\_name>.<permission\_codename> %}*

*Treść z ograniczonym dostępem*

*{% endif %}*

auth\_proj &gt; newapp &gt; templates &gt; newapp &gt; hello.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    {% if perms.newapp.change_grade %}
        Treść dostępna tylko dla użytkowników posiadających
        uprawnienie change_grade.
    {% endif %}

    Treść dostępna dla wszystkich.
</body>
</html>
```

Terminal Local +

```
[12/Apr/2022 14:35:24] "GET /hello/ HTTP/1.1" 200 164
Not Found: /login/
[12/Apr/2022 14:35:31] "GET /login/ HTTP/1.1" 404 2443
[12/Apr/2022 14:35:37] "GET /accounts/login/ HTTP/1.1" 200 729
[12/Apr/2022 14:35:42] "POST /accounts/login/ HTTP/1.1" 302 0
[12/Apr/2022 14:35:42] "GET /hello/ HTTP/1.1" 200 255
[12/Apr/2022 14:35:54] "GET /hello/ HTTP/1.1" 200 164
```

*ZMIENNA PERMS**PRZYKŁAD UŻYCIA*



*GRUPY*

# OPIS

Często rola jaką pełni użytkownik w projekcie związana jest z całym zestawem uprawnień. Najprościej wtedy taki zestaw uprawnień zamknąć w oddzielnym zbiorze. **Groups** (grupy) są właśnie takimi zbiornikami na uprawnienia. Po utworzeniu grupy należy przypisać do niej odpowiednie uprawnienia. Następnie należy przypisać do niej użytkownika (zamiast pojedynczo przypisywać użytkownikowi każde z uprawnień). Użytkownik może należeć do dowolnej liczby grup, a jedna grupa może posiadać dowolną liczbę użytkowników. Po przypisaniu użytkownika do grupy, użytkownik uzyskuje wszystkie uprawnienia zebrane w grupie. W takim scenariuszu weryfikacja dowolnego z zebranych w grupę uprawnień polega na weryfikacji przynależności użytkownika do grupy.

Informacja o przynależności użytkownika do grupy przechowywana jest w atrybucie `groups` modelu użytkownika. Atrybut `groups` pochodzi z domieszki `PermissionMixin` i jest relacją wiele-do-wielu z modelem `Group`.

File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help auth\_proj - C:\Users\jerem\PycharmProjects\auth\_proj\venv\Lib\site-packages\django\contrib\auth\models.py

auth\_proj > venv > Lib > site-packages > django > contrib > auth > models.py

Project External Libraries Scratches and Consoles DB Browser

models.py

```
75 class Group(models.Model):
76
77     name = models.CharField(_("name"), max_length=150, unique=True)
78     permissions = models.ManyToManyField(
79         Permission,
80         verbose_name=_("permissions"),
81         blank=True,
82     )
83
84     objects = GroupManager()
85
86     class Meta:
87         verbose_name = _("group")
88         verbose_name_plural = _("groups")
89
90     def __str__(self):
91         return self.name
```

UserManager

Terminal: Local × +

Performing system checks...

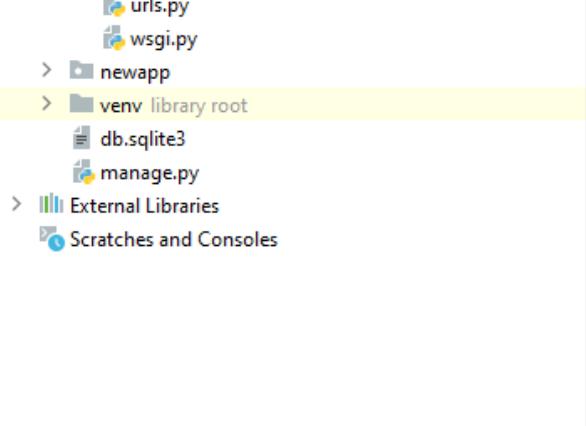
DJANGO.CONTRIB.AUTH.MODELS.GROUP

# PRACA Z MODELEM GROUP

W modelu Group obowiązują wszystkie zasady poznane podczas omawiania Django ORM. Dodatkowo Django posiada narzędzia, których można użyć do weryfikacji przynależności użytkownika do grupy.



*GROUP CRUD*



Search Everywhere Double Shift

Go to File Ctrl+Shift+N

Recent Files Ctrl+E

Navigation Bar Alt+Home

Terminal: Local × + ▾

```
(venv) C:\Users\jerem\PycharmProjects\auth_proj>python manage.py shell
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from django.contrib.auth.models import Group
>>>
>>> teacher_group = Group.objects.create(name='Teacher')
>>> teacher_group
<Group: Teacher>
>>> 
```

Aktywuj system Windows  
Przejdz do ustawieñ, aby aktywowaæ system Windows

*TWORZENIE GRUPY*

object group | Django site X +

→ C 127.0.0.1:8000/admin/auth/group/ 110% ☆

Django administration WELCOME, ADMIN. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Authentication and Authorization > Groups

Start typing to filter...

ACCOUNTS

Users [+ Add](#)

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Select group to change

Action: ----- Go 0 of 1 selected

GROUP

Teacher

1 group

*TWORZENIE GRUPY*

*WIDOK PANELU ADMINISTRATORA*

The screenshot shows the PyCharm IDE interface. On the left, the Project tool window displays a file structure for a Django project named 'auth\_proj' with files like 'accounts.py', 'config/\_init\_.py', 'config/asgi.py', 'config/settings.py', and 'config.urls.py'. Below the project tree is the Terminal window titled 'Local'. The terminal output shows the following Python code:

```
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from django.contrib.auth.models import Group
>>>
>>> teacher_group = Group.objects.create(name='Teacher')
>>> teacher_group
<Group: Teacher>
>>>
>>> from django.contrib.auth.models import Permission
>>> perms = Permission.objects.filter(codename__contains='grade') ← Pobranie uprawnień
>>> perms
<QuerySet [<Permission: newapp | grade | Can add grade>, <Permission: newapp | grade | Can change grade>, <Permission: newapp | grade | Can delete grade>, <Permission: newapp | grade | Can view grade>]>
>>>
>>> teacher_group.permissions.set(perms) ← Przypisanie uprawnień do grupy
>>> █
```

*PRZYPISANIE UPRAWNIEN DO GRUPY*

her | Change group | Django site X +

127.0.0.1:8000/admin/auth/group/1/change/ 110% ☆

Django administration WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Authentication and Authorization > Groups > Teacher

Start typing to filter...

ACCOUNTS

Users + Add

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Change group

**Teacher**

Name: Teacher

Permissions:

Available permissions ? Filter

- auth | group | Can delete group
- auth | group | Can view group
- auth | permission | Can add permission
- auth | permission | Can change permission
- auth | permission | Can delete permission
- auth | permission | Can view permission
- contenttypes | content type | Can add content type
- contenttypes | content type | Can change content type

Chosen permissions ?

- newapp | grade | Can add grade
- newapp | grade | Can change grade
- newapp | grade | Can delete grade
- newapp | grade | Can view grade

PRZYPISANIE UPRAWNIEN DO GRUPY

WIDOK PANELU ADMINISTRATORA

\_proj config

Project auth\_proj C:\Users\jerem\PycharmProjects\auth\_proj

accounts config

  \_\_init\_\_.py  
  asgi.py  
  settings.py  
  urls.py

terminal: Local Local (2) +

```
>> teacher_group.permissions.set(perms)
>>
>> from django.contrib.auth import get_user_model
>> User = get_user_model()
>> bill = User.objects.get(username="Bill")
>>
>> teacher_group.user_set.add(bill) ← Przypisanie użytkownika do grupy
>> █
```

# *PRZYPISANIE UZYTKOWNIKA DO GRUPY*

*(OD STRONY GRUPY)*

\_proj config

Project auth\_proj C:\Users\jerem\PycharmProjects\auth\_proj

accounts config

  \_init\_.py  
  asgi.py  
  settings.py  
  urls.py

terminal: Local Local (2) +

```
>> teacher_group.permissions.set(perms)
>>
>> from django.contrib.auth import get_user_model
>> User = get_user_model()
>> bill = User.objects.get(username="Bill")
>>
>> bill.groups.add(teacher_group) ← Przypisanie użytkownika do grupy
>> █
```

# *PRZYPISANIE UZYTKOWNIKA DO GRUPY*

*(OD STRONY UZYTKOWNIKA)*

Start typing to filter...

## ACCOUNTS

## Users

 Add

## AUTHENTICATION AND AUTHORIZATION

## Groups

 Add

## Permission

 Active

Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

Staff status

Designates whether the user can log into this admin site.

### Superuser status

Designates that this user has all permissions without explicitly assigning them.

## Group

## Available groups

 Filter

Chosen groups ?

Teacher

# *PRZYPISANIE UZYTKOWNIKA DO GRUPY*

## *WIDOK PANELU ADMINISTRATORA*

# *WERYFIKACJA PRZYNALEZNOSCI*

# WERYFIKACJA PRZYNALEZNOSCI DO GRUPY

Wbudowany model użytkownika posiada atrybut **groups**. Wartość atrybutu to wszystkie grupy, do których należy użytkownik. Weryfikacja czy dany użytkownik należy do grupy polega na sprawdzeniu, czy taka grupa znajduje się w wartościach atrybutu groups tego użytkownika. Przykład wywołania:

*bill.groups.filter(name="Teacher").exists()*

lub

*teacher\_group.user\_set.filter(username='Bill').exists()*

Na poziomie widoków przy weryfikacji przynależności użytkownika do grupy możemy posłużyć się dwoma narzędziami:

1. dekorator `user_passes_test`
2. domieszka `UserPassesTestMixin`

Na poziomie szablonów nie istnieje wbudowane narzędzie do weryfikacji przynależności użytkownika do grupy.

Project DB Browser Structure Bookmarks

models.py x

```
203     class PermissionsMixin(models.Model):
204         """
205             Add the fields and methods necessary to support the Group and Permission
206             models using the ModelBackend.
207         """
208
209         is_superuser = models.BooleanField(
210             _("superuser status"),
211             default=False,
212             help_text=_(
213                 "Designates that this user has all permissions without "
214                 "explicitly assigning them."
215             ),
216         )
217         groups = models.ManyToManyField(
218             Group,
219             verbose_name=_('groups'),
220             blank=True,
221             help_text=_(
222                 "The groups this user belongs to. A user will get all permissions granted to each of their groups."
223             ),
224             related_name="user_set",
225             related_query_name="user",
226         )
227         user_permissions = models.ManyToManyField(
228             Permission,
229             verbose_name=_('user permissions'),
230             blank=True,
231             help_text=_("Specific permissions for this user."),
232             related_name="user_set",
233             related_query_name="user",
234         )
235     PermissionsMixin
```

Terminal: Local (2) × Local × Local (3) × + ▾

```
>>> view_grade_perm.user_set.add(student)
>>> █
```

Version Control TODO Problems Python Packages Python Console Terminal

GRUPY  
UZYTKOWNIKA

auth\_proj &gt; newapp &gt; views.py

Project

DB Browser

Structure

Bookmarks

Project

- auth\_proj C:\Users\jerem\PycharmProjects\auth\_proj
  - accounts
  - config
  - newapp
    - migrations
    - templates
    - \_\_init\_\_.py
    - admin.py
    - apps.py
    - models.py
    - tests.py
    - views.py
- > venv library root
  - db.sqlite3
  - manage.py
- > External Libraries
- > Scratches and Consoles

views.py

```

1  from django.urls import reverse_lazy
2  from django.shortcuts import HttpResponseRedirect
3  from django.contrib.auth.decorators import user_passes_test
4
5
6  def is_teacher(user):
7      return user.groups.filter(name='Teacher').exists()
8
9
10 @user_passes_test(is_teacher, login_url=reverse_lazy('my-page'))
11 def update_grade_view(request):
12     return HttpResponseRedirect("Hello!")

```

Terminal: Local (2) × Local × + ▾

```
(venv) C:\Users\jerem\PycharmProjects\auth_proj>python manage.py shell
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
```

(InteractiveConsole)

```

>>> from django.contrib.auth import get_user_model
>>>
>>> User = get_user_model()
>>>
>>> bill = User.objects.get(username="Bill")
>>> bill.groups.filter(name="Teacher").exists()
True
>>>

```

*WERYFIKACJA  
PRZYNALEZNOSCI  
(OD STRONY UZYTKOWNIKA)*

auth\_proj &gt; newapp &gt; views.py

Project

DB Browser

Bookmarks

views.py

```
1 from django.urls import reverse_lazy
2 from django.shortcuts import render
3 from django.views import View
4 from django.contrib.auth.mixins import LoginRequiredMixin
5
6
7 class HelloView(LoginRequiredMixin, View):
8     login_url = reverse_lazy('mv-nade')
```

Terminal Local (2) Local Local (3) +

```
(venv) C:\Users\jerem\PycharmProjects\auth_proj>python manage.py shell
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from django.contrib.auth.models import Group
>>> from django.shortcuts import get_object_or_404
>>>
>>> teacher_group = get_object_or_404(Group, name="Teacher")
>>>
>>> teacher_group.user_set.filter(username="Bill").exists()
True
>>> teacher_group.user_set.filter(username="Sam").exists()
False
>>>
```

*WERYFIKACJA  
PRZYNALEZNOSCI  
(OD STRONY GRUPY)*

# *1. DEKORATOR USER\_PASSES\_TEST*

auth\_proj &gt; newapp &gt; views.py

Project

- auth\_proj C:\Users\jerem\PycharmProjects\auth\_proj
  - accounts
  - config
  - newapp
    - migrations
    - templates
    - \_\_init\_\_.py
    - admin.py
    - apps.py
    - models.py
    - tests.py
  - views.py
- venv library root
  - db.sqlite3
  - manage.py
- External Libraries
- Scratches and Consoles

views.py

```
from django.shortcuts import HttpResponseRedirect
from django.contrib.auth.decorators import user_passes_test

def is_teacher(user):
    return user.groups.filter(name='Teacher').exists()

@user_passes_test(is_teacher)
def update_grade_view(request):
    return HttpResponseRedirect("Hello!")
```

Terminal: Local (2) +

```
[14/Apr/2022 14:03:43] "GET /accounts/login/?next=/hello/ HTTP/1.1" 200 729
C:\Users\jerem\PycharmProjects\auth_proj\newapp\views.py changed, reloading.
Watching for file changes with StatReloader
Performing system checks...
```

System check identified no issues (0 silenced).

April 14, 2022 - 14:31:36

Django version 4.0.3, using settings 'config.settings'  
Starting development server at <http://127.0.0.1:8000/>  
Quit the server with CTRL-BREAK.

USER\_PASSES\_TEST

auth\_proj &gt; newapp &gt; views.py

Project

auth\_proj C:\Users\jerem\PycharmProjects\auth\_proj

- > accounts
- > config
- newapp
  - > migrations
  - > templates
  - \_\_init\_\_.py
  - admin.py
  - apps.py
  - models.py
  - tests.py
  - views.py

> venv library root

- db.sqlite3
- manage.py

> External Libraries

Scratches and Consoles

views.py

```
from django.urls import reverse_lazy
from django.shortcuts import HttpResponseRedirect
from django.contrib.auth.decorators import user_passes_test

def is_teacher(user):
    return user.groups.filter(name='Teacher').exists()

@user_passes_test(is_teacher, login_url=reverse_lazy('my-page'))
def update_grade_view(request):
    return HttpResponseRedirect("Hello!")
```

Terminal: Local (2) +

Quit the server with CTRL-BREAK.

C:\Users\jerem\PycharmProjects\auth\_proj\newapp\views.py changed, reloading.

Watching for file changes with StatReloader

Performing system checks...

System check identified no issues (0 silenced).

April 14, 2022 - 14:32:43

Django version 4.0.3, using settings 'config.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CTRL-BREAK.

*USER\_PASSES\_TEST**PARAMETR LOGIN\_URL*

## *2. DOMIESZKA UserPASSESTESTMIXIN*

auth\_proj &gt; newapp &gt; views.py

Project

DB Browser

Structure

Bookmarks

Version Control

TODO

Problems

Python Packages

Python Console

Terminal

Project

+ -

Settings

auth\_proj C:\Users\jerem\PycharmProjects\auth\_proj  
  > accounts  
  > config  
  > newapp  
    > migrations  
    > templates  
    > \_\_init\_\_.py  
    > admin.py  
    > apps.py  
    > models.py  
    > tests.py  
    > views.py  
  > venv library root  
    db.sqlite3  
  > manage.py  
> External Libraries  
Scatches and Consoles

views.py

```
1  from django.shortcuts import render
2  from django.views import View
3  from django.contrib.auth.mixins import UserPassesTestMixin
4
5
6  def is_teacher(user):
7      return user.groups.filter(name='Teacher').exists()
8
9
10 class UpdateGradeView(UserPassesTestMixin, View):
11
12     def test_func(self):
13         return is_teacher(self.request.user)
14
15     def get(self, request):
16         return render(
17             request,
18             'newapp/hello.html'
19         )
20
21
```

Terminal: Local (2) × Local + ▾

&gt;&gt;&gt; bill.groups.filter(name="Teacher").exists()

True

&gt;&gt;&gt; █

USERPASSESTESTMIXIN

auth\_proj &gt; newapp &gt; views.py

Project DB Browser Structure Bookmarks

views.py

```
from django.shortcuts import render
from django.views import View
from django.contrib.auth.mixins import UserPassesTestMixin

def is_teacher(user):
    return user.is_teacher

class UpdateView(UserPassesTestMixin, View):
    def test_func(self):
        return self.request.user.is_teacher()

    def get(self, request, *args, **kwargs):
        return render(request, 'newapp/hello.html', {'name': 'Hello World'})
```

403 Forbidden

Terminal: Local (2) Local + <

```
>>> bill.groups.filter(name="Teacher").exists()
True
>>>
```

USERPASSESTESTMIXIN

auth\_proj &gt; newapp &gt; views.py

```
from django.shortcuts import render, redirect
from django.views import View
from django.contrib.auth.mixins import UserPassesTestMixin

def is_teacher(user):
    return user.groups.filter(name='Teacher').exists()

class UpdateGradeView(UserPassesTestMixin, View):
    def test_func(self):
        return is_teacher(self.request.user)

    def handle_no_permission(self):
        return redirect('my-page')

    def get(self, request):
        return render(
            request,
            'newapp/hello.html'
        )
```

*USERPASSESTESTMIXIN**PRZEKIEROWANIE*

## 4. ROLE VS GRUPY

Może wydawać się, że role i grupy pełnią tę samą funkcję. W kodzie ich funkcje jednak różnią się. Zaleca się aby w projekcie używać obu. Ról w logice biznesowej aplikacji, a grup do zarządzania uprawnieniami (udzielaniu lub blokowaniu dostępu do strony, wyświetlanu lub nie fragmentów strony).

Dobrą praktyką jest tworzenie nowej grupy dla każdej nowo utworzonej roli. Razem z rozwojem projektu należy rozszerzać istniejące grupy o nowo powstałe uprawnienia.



# *WBUDOWANY MODEL UZYTKOWNIKA*

INFORMACJE DODATKOWE

# 1. ATRYBUT USERNAME JEST CZULY NA WIELKOSC LITER

Atrybut username ma unikalną wartość (parametr unique ma wartość True), ale jest czuły na wielkość liter. Oznacza to, że użytkownik o nazwie john i John będą traktowani jako dwa odrębne konta.

## 2. ATRYBUT USERNAME AKCEPTUJE ZNAKI UNICODE

Atrybut username domyślnie akceptuje litery, liczby oraz znaki: '@', '.', '+', '-', '\_'. Pułapką jest tutaj termin litery. Domyślny validator UnicodeUsernameValidator akceptuje litery z kodowania Unicode. Oznacza to, że nazwy użytkowników: joão, Джон, a nawet 約翰 to poprawne nazwy.

### 3. POLE EMAIL NIE JEST UNIKALNE

Domyślnie wielu użytkowników może mieć tą samą wartość w polu email. Jest to o tyle kłopotliwe, że email jest używany m.in. do odzyskiwania hasła. Poza tym wiele portali używa email-a do uwierzytelnienia użytkownika.

## 4. POLE EMAIL NIE OBOWIAZKOWE

Co prawda pole email nie może przyjmować pustej wartości (None w Pythonie, null na bazie), ale domyślnie można zapisać w kolumnie email pusty string.

# *DODATKI*



SERWER



Klient

*ARCHITEKTURA  
Klient-Serwer*