

DOKUMENTACJA TO-DO-APP

PROJEKT PORTFOLIO PAWEŁ FIUK

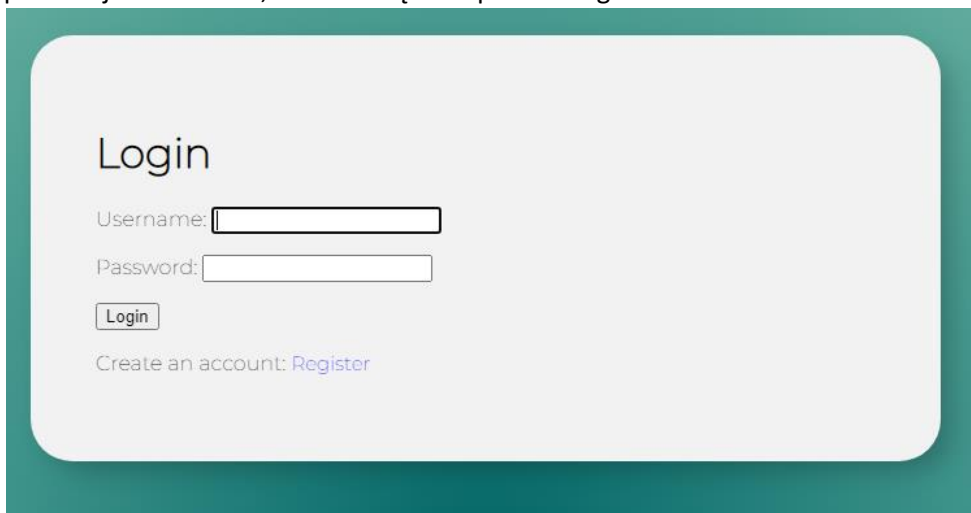
GITHUB: github.com/PawelFiuk/To-do-app

1. CZYM JEST PROJEKT TO-DO-APP?

Projekt jest aplikacją internetową wykonaną za pomocą języka Python oraz frameworka Django. W tej aplikacji można tworzyć własne zadania do wykonywania, odczytać zawartość, zmieniać je, usuwać (CRUD). Użytkownik może stworzyć własne konto, zalogować i wylogować się. Po zalogowaniu użytkownik widzi tylko i wyłącznie swoje zadania, może je edytować i usuwać.

2. JAK KORZYSTAĆ Z APLIKACJI?

Po wejściu do aplikacji użytkownik zostanie przekierowany do strony z logowaniem. Jeżeli nie posiada jeszcze konta, musi kliknąć w hiperlink Register.



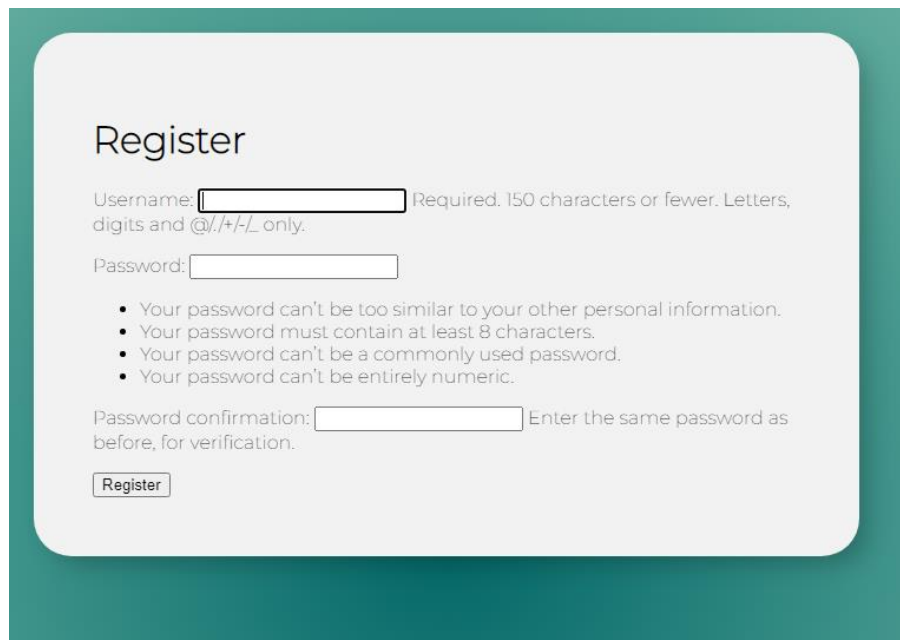
Login

Username:

Password:

Create an account: [Register](#)

Potem zostanie przekierowany do strony z formularzem stworzenia nowego konta. Należy wykonać czynności zgodnie z instrukcjami zamieszczonymi na stronie.

A registration form titled "Register" on a teal background. It includes input fields for "Username:" and "Password:", with a "Password confirmation:" field below. A "Register" button is at the bottom. A list of password requirements is shown: "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.", and "Your password can't be entirely numeric." The form is enclosed in a light gray rounded rectangle.

Register

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

Password:

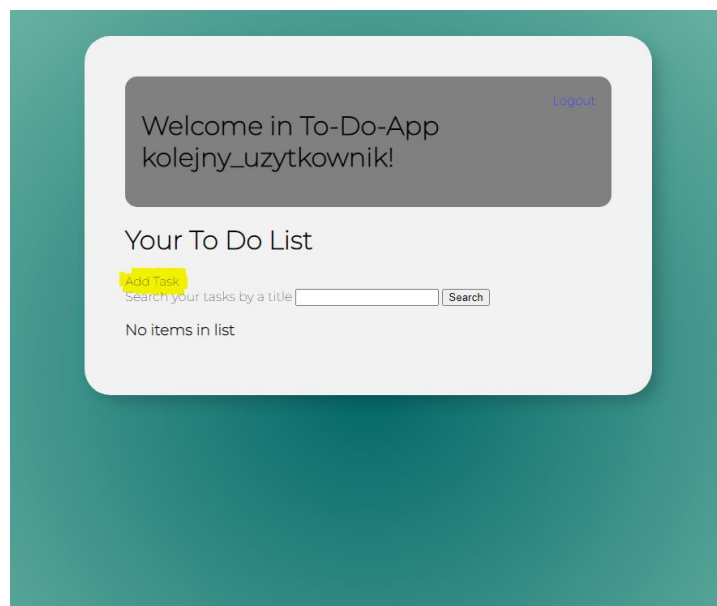
- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

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

Po stworzeniu konta, użytkownik zostanie od razu zalogowany i przeniesiony do strony głównej, gdzie znajdują się zadania. Jeżeli nie ma żadnych zadań, wyświetli się napis *No tasks to complete. Congratulations!*

Tworzenie nowego zadania (task):

Żeby utworzyć nowe zadanie, należy kliknąć napis Add task (zaznaczony na żółto na obrazku)

The home screen of the "To-Do-App" on a teal background. It features a "Welcome in To-Do-App" message with a "Logout" link. Below is a "Your To Do List" section with a yellow "Add Task" button, a search bar with a "Search" button, and the text "No items in list".

Welcome in To-Do-App
kolejny_uzytkownik! [Logout](#)

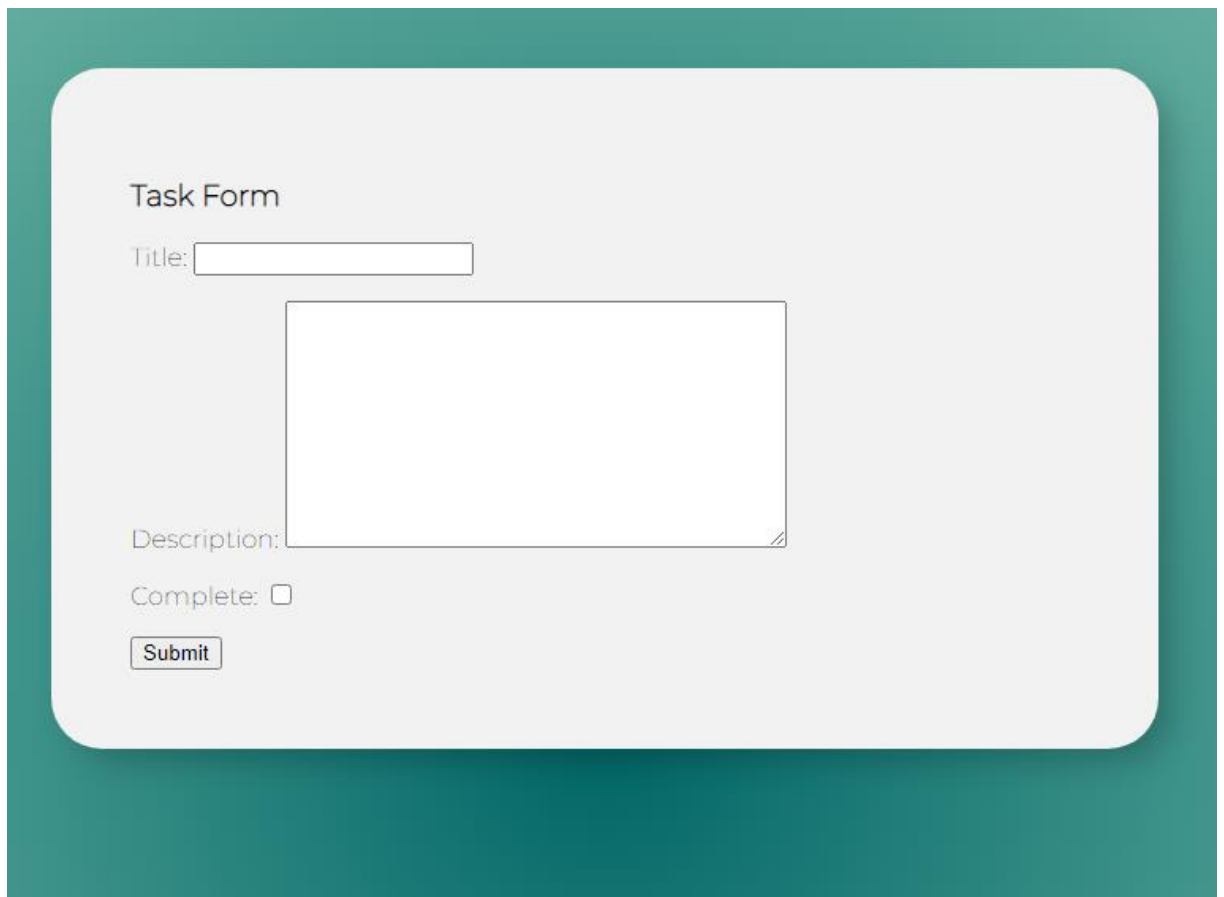
Your To Do List

[Add Task](#)

Search your tasks by a title

No items in list

Po tym wydarzeniu użytkownik zostanie odesłany do strony z formularzem zadania. Jest tutaj kilka pól do wypełnienia: Title (musi zostać wypełniony) – tytuł zadania, Description -opis zadania , Complete – checkbox, który wskazuje na to czy zadanie zostało wykonane. Po wykonaniu zadania należy wcisnąć przycisk Submit, który wyśle formularz do bazy danych.

A screenshot of a web form titled "Task Form" on a teal background. The form is contained within a light gray rounded rectangle. It features a "Title:" label followed by a text input field. Below this is a "Description:" label followed by a larger text area. At the bottom left of the form is a "Complete:" label with an unchecked checkbox. At the bottom center is a "Submit" button.

Schemat formularza jest zapisany w aplikacji base pliku models.py, w klasie Task, która dziedziczy po klasie models.Model, pozwalającej na tworzenie modeli

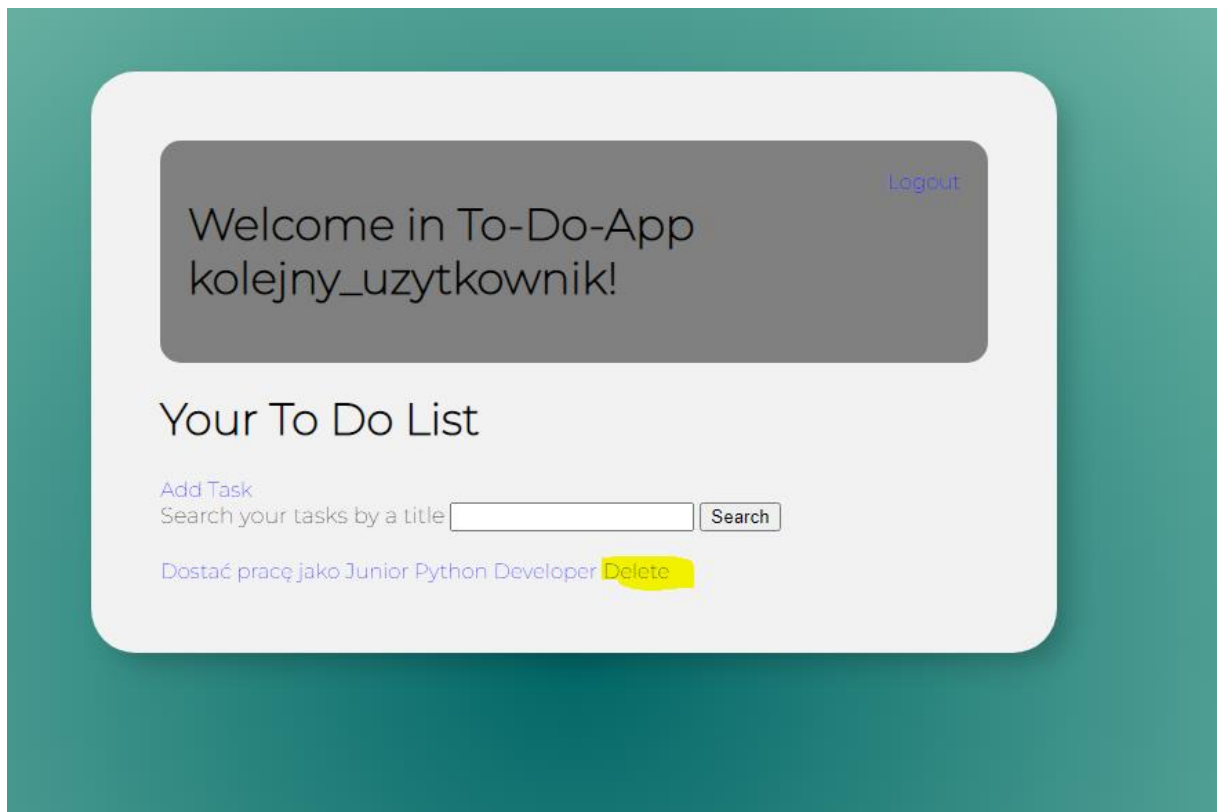
```
from django.db import models
from django.contrib.auth.models import User

class Task(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE, null=True, blank=True)
    title = models.CharField(max_length=200)
    description = models.TextField(null=True, blank=True)
    complete = models.BooleanField(default=False)
    create = models.DateTimeField(auto_now_add=True)

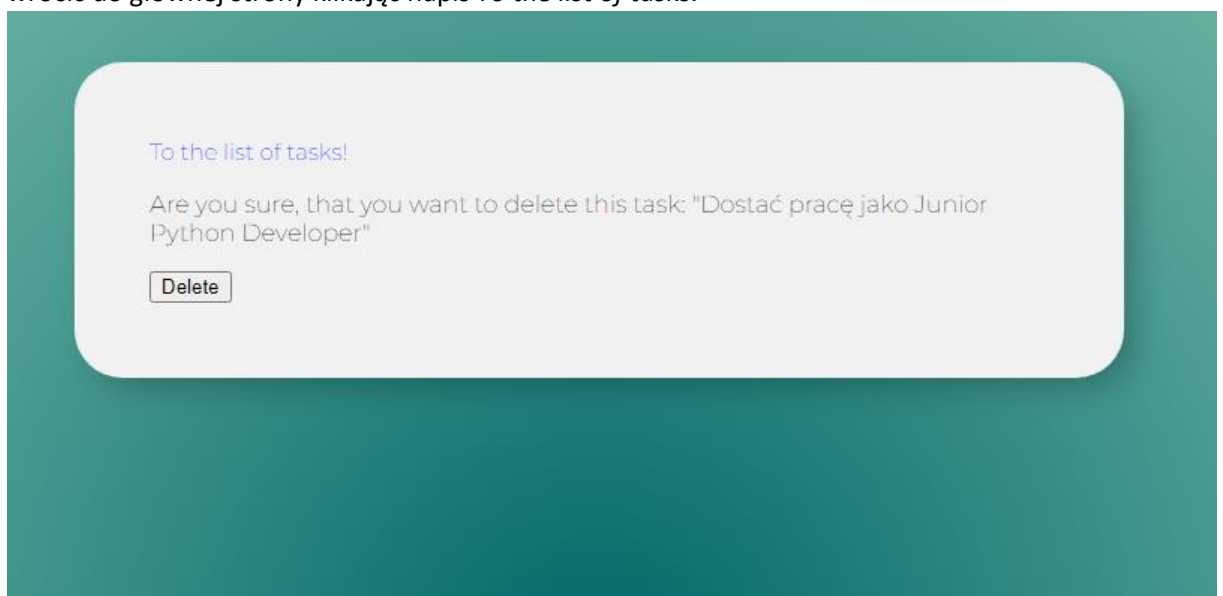
    def __str__(self):
        return self.title

    class Meta:
        ordering = ['complete']
```

Po utworzeniu zadania, tytuł wyświetli się na głównej stronie. Możliwe jest wejście i modyfikacja zadania, klikając na jego tytuł. Zadanie można też usunąć klikając napis Delete.



Po tym wydarzeniu użytkownik zostanie przekierowany do strony w której zostanie zapytany czy na pewno chce usunąć dane zadanie. Po usunięciu zadania, użytkownik zostanie ponownie przekierowany do strony głównej. Użytkownik może jednak nie usuwać zadania i wrócić do głównej strony klikając napis *To the list of tasks*.



3. PLIKI W PROJEKCIE

a) W aplikacji base, w pliku views.py znajdują się klasy widoków.

Zawartość pliku:

-class CustomLoginView(LoginView) – służy do stworzenia widoku logowania się,
template html tego logowania znajduje się w base/login.html. Widok pobiera

wszystkie wartości z modelu Task w zmiennej fields, po zalogowaniu użytkownik jest przekierowywany do strony głównej 'task' przez funkcję get_success_url

-class RegisterPage(FormView) – służy do utworzenia formularza do stworzenia konta. W funkcji form_valid jeżeli wszystkie pola w formularzu zostaną wypełnione prawidłowo, konto zostanie zapisane w bazie danych.

-class TaskList(LoginRequiredMixin, ListView) - służy do wyświetlenia zadań użytkownika. Dziedziczy po klasach LoginRequiredMixin (która odpowiada za autoryzację, czy użytkownik jest zalogowany) i ListView (Która służy do odczytu z bazy danych naszego zadania). W funkcji get_context_data zaimplementowane jest wyszukiwanie zadań po tytule. W zmiennej model jest wybrany model Task, ten sam który znajduje się w pliku models.py. Model ten jest taki sam dla wszystkich widoków poniżej

-class TaskDetail(LoginRequiredMixin, DetailView) – służy do zobaczenia szczegółów zadania. Użytkownik może zobaczyć tylko swoje zadania dzięki dziedziczeniu z LoginRequiredMixin.

-class TaskCreate(LoginRequiredMixin, ListCreate) – służy do stworzenia nowego zadania., Użytkownik może stworzyć swoje zadania dzięki dziedziczeniu z LoginRequiredMixin.

-class TaskUpdate(LoginRequiredMixin, ListUpdate) – służy do zmiany stworzonego wcześniej zadania. Użytkownik może zmienić tylko swoje zadania dzięki dziedziczeniu z LoginRequiredMixin.

-class TaskDelete(LoginRequiredMixin, ListUpdate) – służy do usunięcia danego zadania. Użytkownik może usunąć tylko swoje zadania dzięki dziedziczeniu z LoginRequiredMixin.

b) W aplikacji base w pliku urls.py znajdują się wszystkie linki URL tej aplikacji. Później są one podpięte do głównego pliku urls.py w aplikacji to_do_app za pomocą metody include.

```

from django.urls import path
from .views import TaskList, TaskDetail, TaskCreate, TaskUpdate, TaskDelete, C
from django.contrib.auth.views import LogoutView

urlpatterns = [
    path('', TaskList.as_view(), name='tasks'),
    path('login/', CustomLoginView.as_view(), name='login'),
    path('logout/', LogoutView.as_view(next_page='login'), name='logout'),
    path('task/<int:pk>', TaskDetail.as_view(), name='task'),
    path('task-create', TaskCreate.as_view(), name='tasks-create'),
    path('task-update/<int:pk>', TaskUpdate.as_view(), name='tasks-update'),
    path('task-delete/<int:pk>', TaskDelete.as_view(), name='tasks-delete'),
    path('register/', RegisterPage.as_view(), name='register'),
]

```

' ' – jest do główny url, którego nazwa w zmiennej name='task'

'login/' – url który służy do logowania, wykorzystuje klasę widoku CustomLoginView i metodę .as_view(), pozwalającą na włączenie tej klasy. Metoda ta będzie wykorzystywana we wszystkich poniższych widokach.

'logout/' - url służący do wylogowania się, wykorzystuje klasę widoku LogoutView

'task/<int:pk>' -url służący do zobaczenia treści zadania, w parametrze <int:pk> bierze numer id z bazy danych wybranego zadania, np. pierwsze zadanie w bazie danych będzie miało url 'task/1' itd. Zapis ten będzie miał takie samo działanie jak w poniższych przykładach

'task-create<int:pk>' – url służący do stworzenia nowego zadania. Wykorzystuje klasę widoku TaskUpdate

'task-delete/<int:pk>' – url służący do usunięcia zadania. Wykorzystuje klasę TaskDelete.

'register' – url służący do stworzenia konta użytkownika. Wykorzystuje klasę widoku RegisterPage

c) Folder z template plików html znajduje się w aplikacji base, w folderze base/template

Najważniejszym templatem jest base_layout.html, to z niego za pomocą tagu Django

{% extends %} i {% block content %} pseudo-dziedziczą pozostałe pliki html w aplikacji. Zamieszczone są metadane oraz jest podpięty plik style.css służący do stylizacji strony, plik ten znajduje się w folderze base/static.

Plik login.html zawiera strukturę html do zalogowania się użytkownika, znajduje się tam formularz do zalogowania zabezpieczony tokenem csrf_token, wykorzystuje klasę widoku CustomLoginView.

Plik register.html zawiera strukturę html z formularzem służącym do stworzenia konta, wykorzystuje klasę widoku RegisterPage

Plik task_confirm_delete.html zawiera strukturę html służącą do potwierdzenia usunięcia danego zadania. Wykorzystuje widok TaskDelete.

Plik task_detail.html zawiera strukturę html służącą do zobaczenia opisu zadania, wykorzystuje klasę widoku TaskDetail

Plik task_form.html zawiera strukturę html służącą do stworzenia nowego zadania. Znajduje się tam formularz zabezpieczony tokenem csrf i przycisk Submit. Wykorzystuje klasę widoku TaskCreate

Plik task_list.html to główna strona aplikacji, w niej są pokazane wszystkie zadania, są przyciski służące do wylogowania się, stworzenia nowego zadania i usunięcia go. Wykorzystuje klasę widoku TaskList.