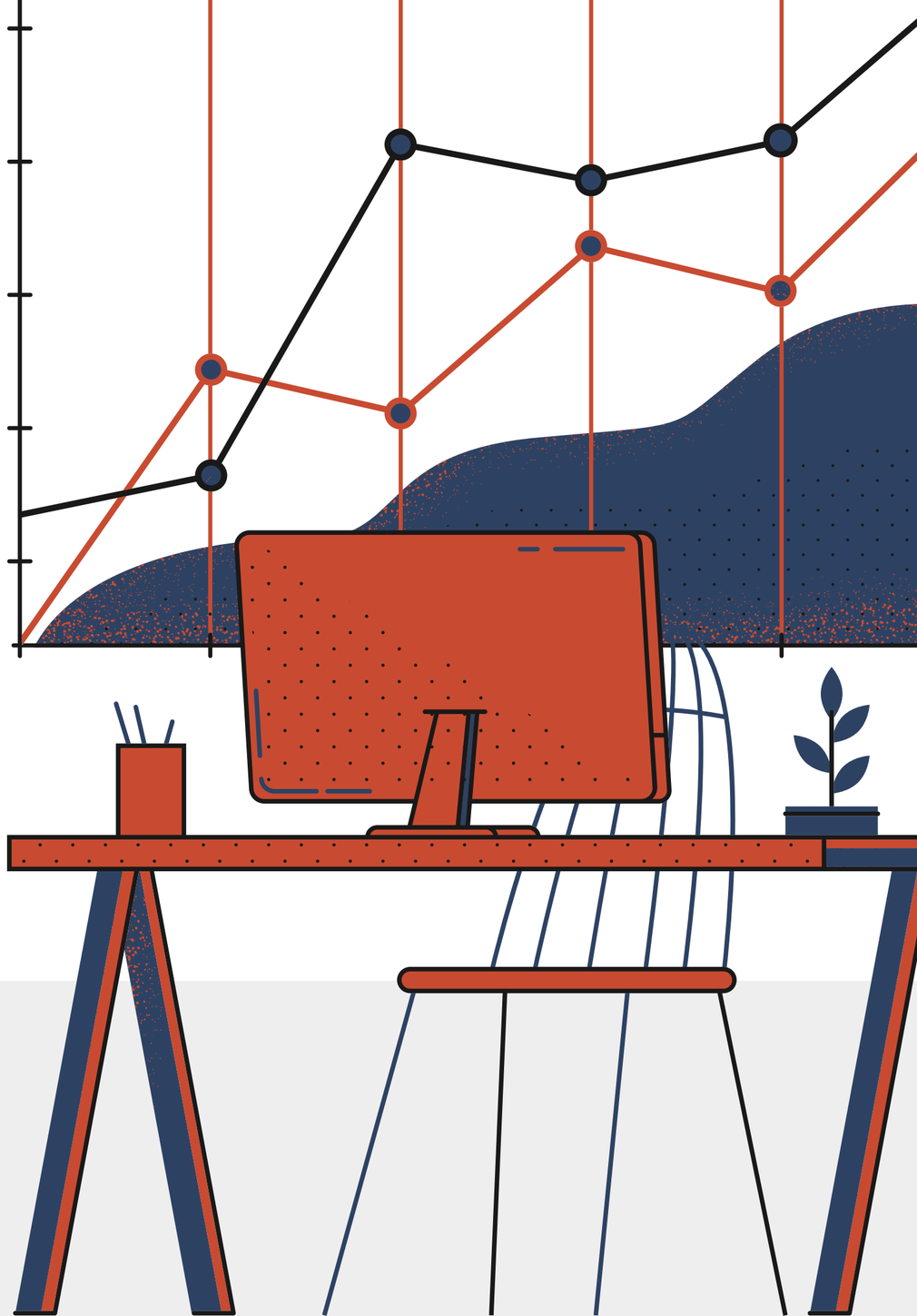
2026 January 9

Django, Flask, FastAPI



AUTHOR

Paulina Kulczyk

SUPERVISOR

Łukasz Lewoń

Table of Contents

Some welcome words

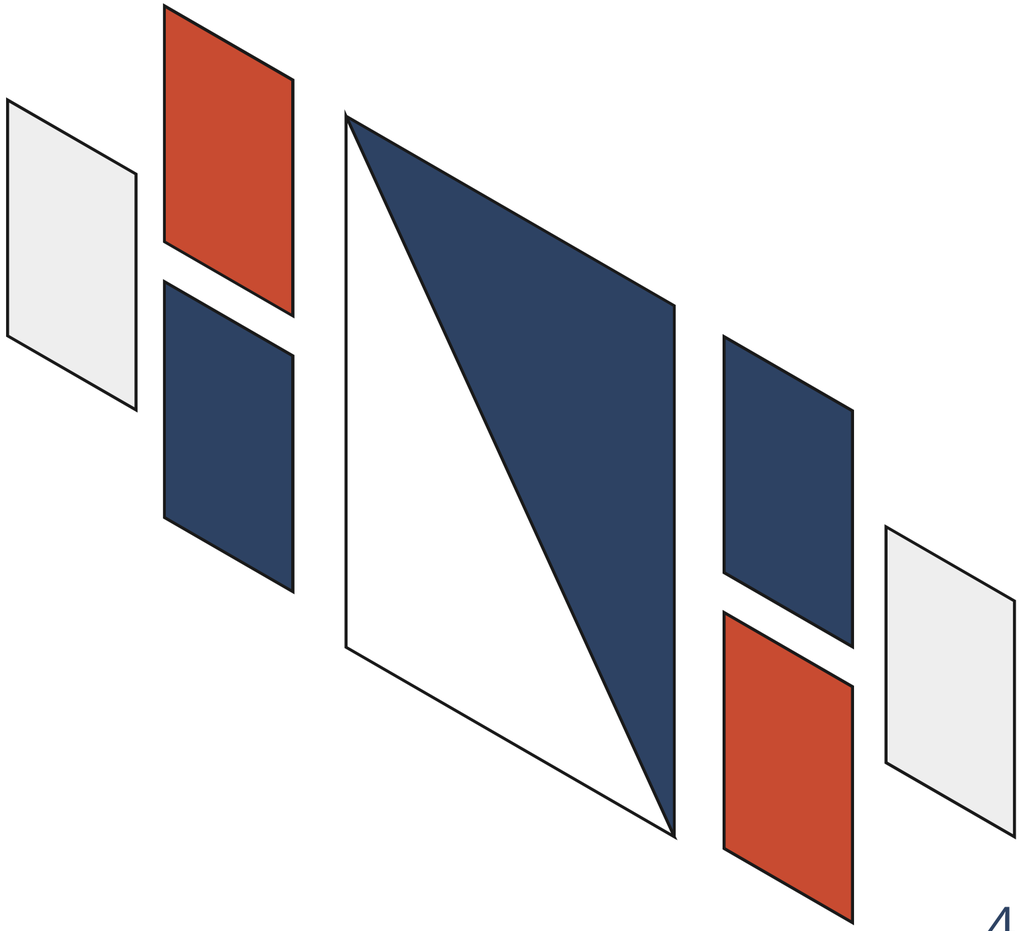
- **Django**: A full-stack web framework that comes with **“batteries included” featuring, a built-in ORM, an admin interface**, and following the MVT (Model-View-Template) architecture.
- **Flask**: A lightweight and flexible micro-framework that allows developers to choose their tools, using Jinja2 for templates and supporting many extensions.
- **FastAPI**: A modern, high-performance framework designed for asynchronous programming, leveraging Python type hints and providing automatic API documentation.

Admin panel - how I will be managing my app?

Django	Flask	FastAPI
Comes with a built-in, fully-featured, ready to use admin interface for managing models, users, and data out-of-the-box.	**No built-in admin panel**, but you **can use extension like Flask-Admin**. More setup and customization required, escpesially for big apps with different ‘subapps’.	**No built-in admin panel.** You can build it using for example **sqladmin but then you need also other tools** for handling forms and then use also tables to authorization but then you have to have **lots of code** with dependencies and then with frontend.
Built-in authentication system, including user models, permissions, groups, and session management. Support Supports **many-to-many relationships (users ↔ groups) and cascading/hierarchical roles**. Easy to extend for complex authorization.	Has **extensions like Flask-Login and Flask-Security** for authentication and role-based authorization.Requires coding and for handle more complex you have to built by yourself. **Cascading groups must be implemented manually**	**No built-in auth system**, but supports OAuth2, JWT, and custom security schemes through dependencies.

UI - can I build it without additional libraries?

Django	Flask	FastAPI
Have special html django templates. It helps to handle logic and forms.	Need installation of Jinja - extender for dynamic HTML, but it doesn't support forms handling - for this we need write own code or use other extension	Need installation of Jinja - extender for dynamic HTML, but it doesn't support forms handling - for this we need write own code or use other extension
For bigger UI is best to use different UI library and create REST API connection.	For bigger UI is best to use different UI library and create REST API connection	For bigger UI is best to use different UI library and create REST API connection



Society words

Mówi się, że Django jest bardzo dobre do zarządzania autoryzacja i autentykacja, ma gotowy już panek admina (wiadomo trzeba go rozbudowywać ale jest w pakiecie w paczce, pozostałe frameworki dowiedziałam się, że tak nie maja, do flaska chyba jest dostępna biblioteka do tworzenia panelu admina)

Django jest mainstreamowym frameworkiem – standardem. Wiele osób ma z tym problem i próbują użyć czegoś innego tylko dlatego, że Django jest super popularne.
Sam nie jestem bez winy, bo w 2012 roku wybierałem wszystkie inne frameworki tylko nie Django. Myślałem wtedy, że będzie mnie ograniczać. Przypadkowo wskoczyłem do istniejącego projektu w Django i otworzyło mi to oczy. Wszystko było w nim zorganizowane tak, jak powinno. Byłem o wiele bardziej produktywny i mogłem skupić się na rzeczywistych problemach klienta zamiast walczyć z frameworkiem lub pisać kod który ktoś już kiedyś napisał.

- *Źródło: https://justjoin.it/blog/flask-vs-django-dlaczego-nie-warto-wybierac-flaska-w-2021-roku*

Django

Starting new project

`django-admin startproject mysite`

- This command creates for us mysite app directory, with:
  - **manage.py**: A command-line utility that lets you interact with this Django project in various ways.
  - **mysite/**: A directory that is the actual Python package for your project.

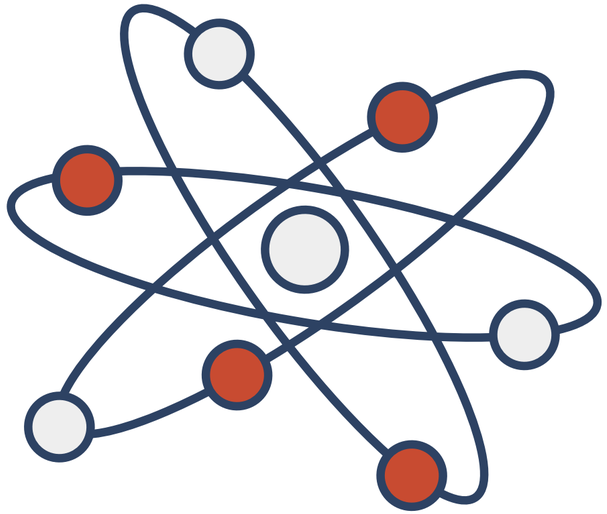
`python manage.py runserver`

- This command starts your first app

```
djangotutorial/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        asgi.py
        wsgi.py
```

The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.



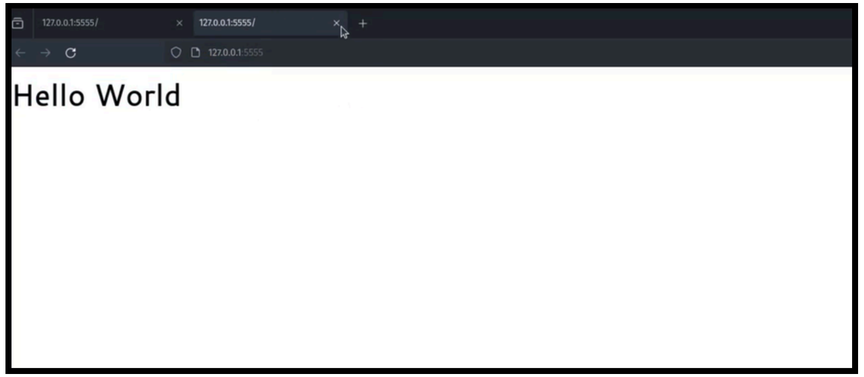
Flask

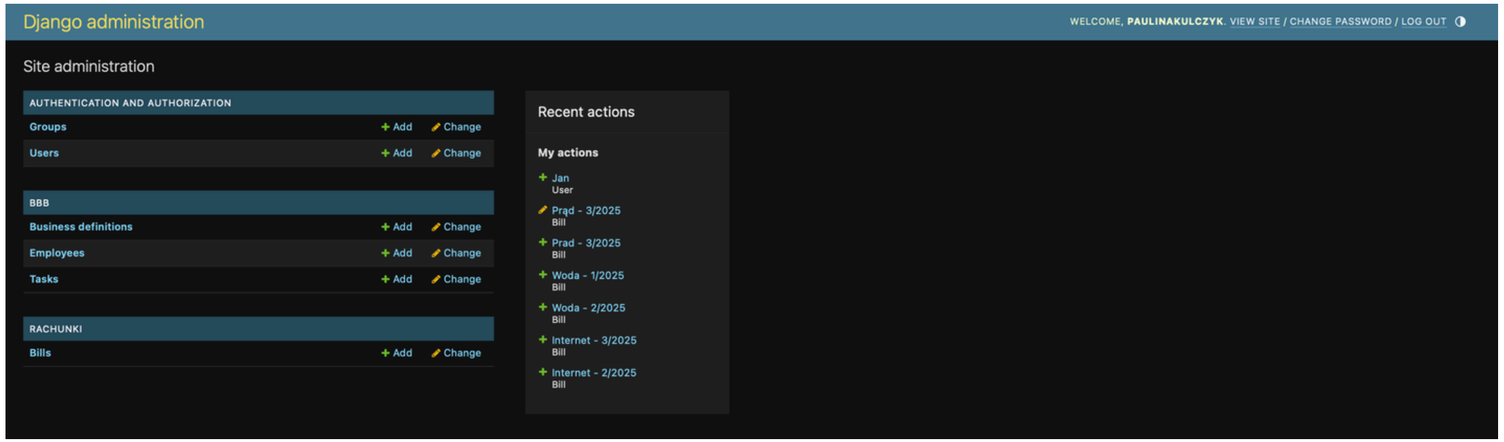
with Flask we don’t have any structure. We have to start from scratch!!!

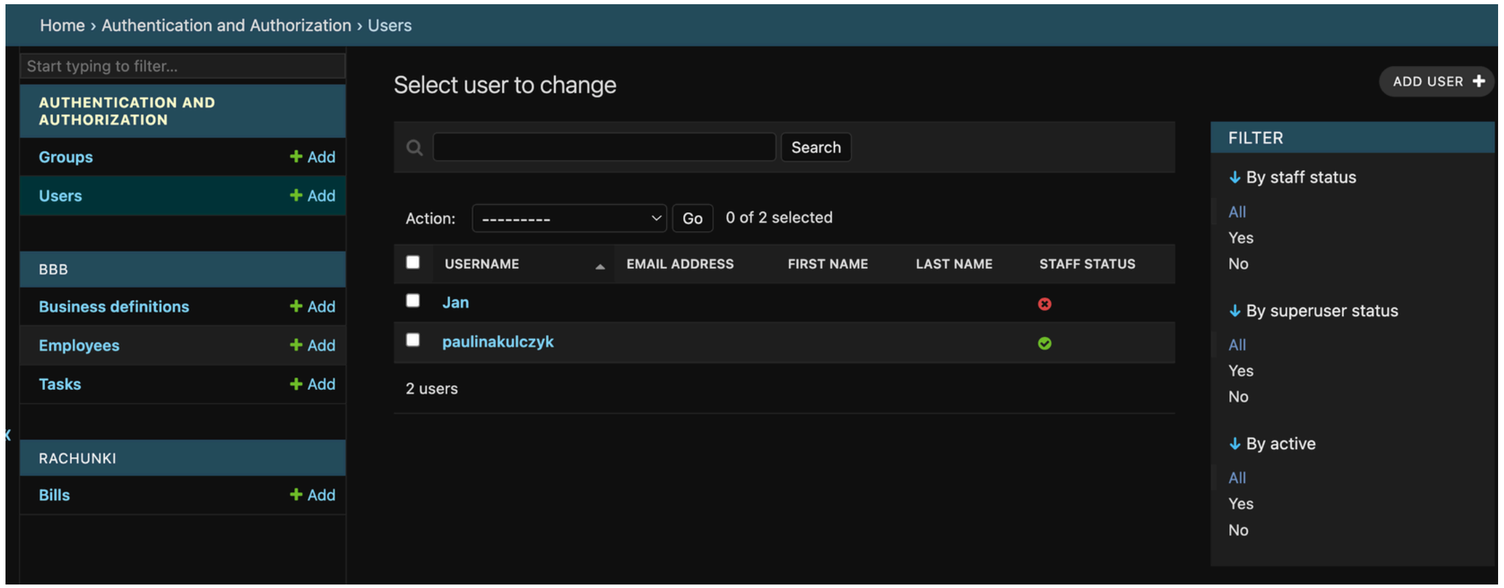
Example:

```python
from flask import Flask

app = Flask(__name__)


@app.route('/')
def index():
    return "<h1>Hello World</h1>"


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5555, debug=True)
```

app.py







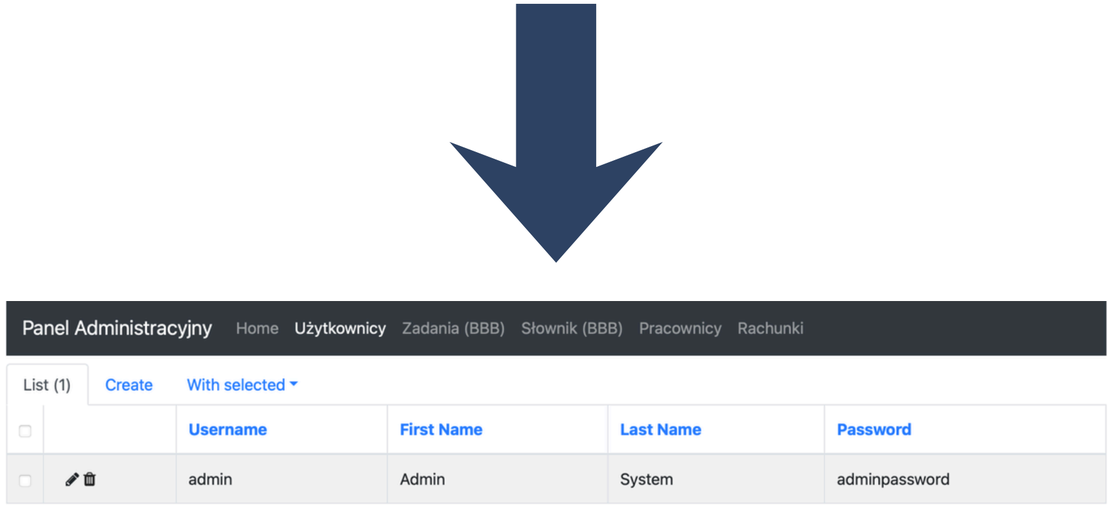
We get admin panel from scratch!

When we run *django-admin startproject mysite* command we get admin panel already.

In this admin panel we see three sections:

- Authorization & grups ← have from beginning, here we can manage the users rights and add other users (we can also create users with LDAP etc)
- BBB ← section that was created by django when we created BBB subapp (the same with Rachunki)

```python
# --- ADMIN PANEL ---

admin = Admin(app, name='Panel Administracyjny')

# Views - they generate form in our UI admin panel, they handle all the CRUD logic
admin.add_view(ModelView(User, db.session, name="Użytkownicy"))
admin.add_view(ModelView(Task, db.session, name="Zadania (BBB)"))
admin.add_view(ModelView(BusinessDefinition, db.session, name="Słownik (BBB)"))
admin.add_view(ModelView(Employee, db.session, name="Pracownicy"))
admin.add_view(ModelView(Bill, db.session, name="Rachunki"))
```



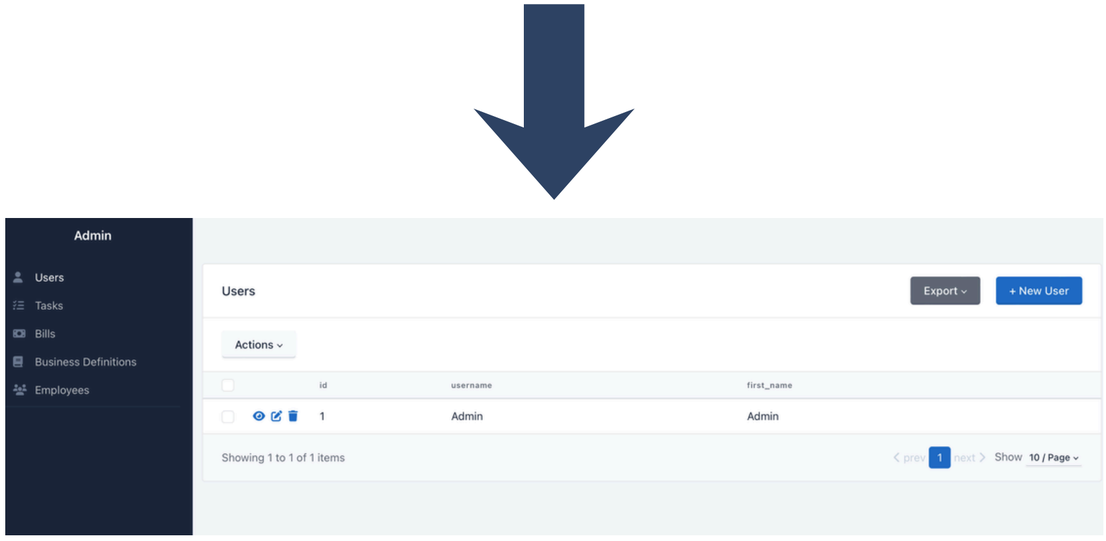
In Flask we don't have from default admin panel but we can use *flask_admin* library

Here one of the most important component is **ModelView from flask_admin** - a bridge between your database models and the browser form.

As we see here, with code we generate basic users administration panel, but we don't have authorization options as in Django. Belowe the code for this basic users panel:

class User(db.Model):
id = db.Column(db.Integer, primary_key=True)
username = db.Column(db.String(80), unique=True, nullable=False)
first_name = db.Column(db.String(100))
last_name = db.Column(db.String(100))
password = db.Column(db.String(200))

```python
# --- ADMIN PANEL CONFIG (SQLAdmin) ---
class UserAdmin(ModelView, model=User):
    column_list = [User.id, User.username, User.first_name]
    icon = "fa-solid fa-user"

class TaskAdmin(ModelView, model=Task):
    column_list = [Task.title, Task.due_date]
    form_columns = [Task.title, Task.description, Task.due_date, Task.assigned_to]
    icon = "fa-solid fa-list-check"

class BillAdmin(ModelView, model=Bill):
    column_list = [Bill.category, Bill.amount, Bill.date]
    icon = "fa-solid fa-money-bill"

class DefAdmin(ModelView, model=BusinessDefinition):
    column_list = [BusinessDefinition.term]
    icon = "fa-solid fa-book"
```



In FastApi we handle the same problem - no admin panel, no authentication and authorization schema

Here once again we use the **ModelView but this time imported from sqladmin** - a bridge between your database models and the browser form. With this model we aslo build users. To authentication we use bcrypt and fastapi.security library.

It works similar to flask_admin ModelView so we don’t present here code.

2026 January 9

Thank You for Listening