

Building an Admin Panel for FastAPI Apps | Step-by-Step Tutorial | Sqladmin



Shubhendu ghosh · [Follow](#)

3 min read · Jun 11, 2024



Listen



Share



More



Photo by [Stanley Dai](#) on [Unsplash](#)

Introduction

FastAPI, known for its lightning-fast performance and intuitive design, has emerged as a favorite among developers seeking to streamline their backend processes. Today in this article we will see how can we create an admin panel. This is a complete step-by-step tutorial will cover everything you need to build a feature-rich admin interface from scratch. Let's

harness the power of FastAPI and embark on a journey to build an admin panel that not only meets but exceeds your expectations. Let's dive in!

Step -1 : Installation

Before we start let's clear the dependencies. we will use *sqlalchemy* for database operation and *sqladmin* for admin service. These two are very important because if you are using any other database operation library or other admin then this article might not help you.

```
pip install sqlalchemy
pip install sqladmin
pip install sqladmin[full]
```

Step-2 : Create the Database

create a `example_database.py` file and create or link your database with sqlalchemy

```
from sqlalchemy import create_engine
from sqlalchemy.orm import declarative_base
from sqlalchemy.orm import sessionmaker

engine = create_engine("sqlite:///example.db", isolation_level='AUTOCOMMIT')

session = sessionmaker(bind=engine)
Base = declarative_base()
```

[Open in app](#) ↗

Medium

 Search



```
from sqlalchemy import Column, Integer, String, Boolean
from example_database import Base, engine, session
```

```
class User(Base):
    __tablename__ = "users"

    id = Column(Integer, primary_key=True)
    email = Column(String)
```

```
username = Column(String)
password = Column(String)
firstname = Column(String)
lastname = Column(String)
is_admin = Column(Boolean)
```

```
Base.metadata.create_all(engine)
```

Step-4: Create main.py

Now create a file named main.py. This will be the main file of your fastapi app.

```
from fastapi import FastAPI
from admin import create_admin #we will create this file in a while

app = FastAPI(title="AppName")
admin = create_admin(app) # we will write this function in a while

# list your APIs here
@app.get("/")
async def hello_world():
    return {"message": "Hello World"}
```

Step-5: Create admin.py

Now create another file named admin.py. all the admin actions will be controlled from here. Also authentication will be handled from here.

```
from sqlalchemy import Admin, ModelView
from example_database import engine, session
from models.py import Users
from sqlalchemy.authentication import AuthenticationBackend
from starlette.requests import Request

#This page will implement the authentication for your admin pannel
class AdminAuth(AuthenticationBackend):

    async def login(self, request: Request) -> bool:
        form = await request.form()
        username= form.get("username")
        password= form.get("password")
        session = session()
        user = session.query(Users).filter(Users.username == username).first()
        if user and password== user.password:
            if user.is_admin:
```

```

        request.session.update({"token": beta_user.username})
        return True
    else:
        False

    async def logout(self, request: Request) -> bool:
        request.session.clear()
        return True

    async def authenticate(self, request: Request) -> bool:
        token = request.session.get("token")
        return token is not None

# create a view for your models
class UsersAdmin(ModelView, model=Users):
    column_list = [
        'id', 'email', 'first_name', 'last_name', 'is_admin'
    ]

# add the views to admin
def create_admin(app):
    authentication_backend = AdminAuth(secret_key="supersecretkey")
    admin = Admin(app=app, engine=engine, authentication_backend=authentication_backend)
    admin.add_view(UsersAdmin)

    return admin

```

That's it . we are finished with the coding. now just a few steps are left.

Step-6 : Migrate your models to database

run this command to create a migration script.

```
alembic revision --autogenerate -m "Add table Users"
```

Then apply the migration to move your model to your database

```
alembic upgrade head
```

Step-7 : Run the app

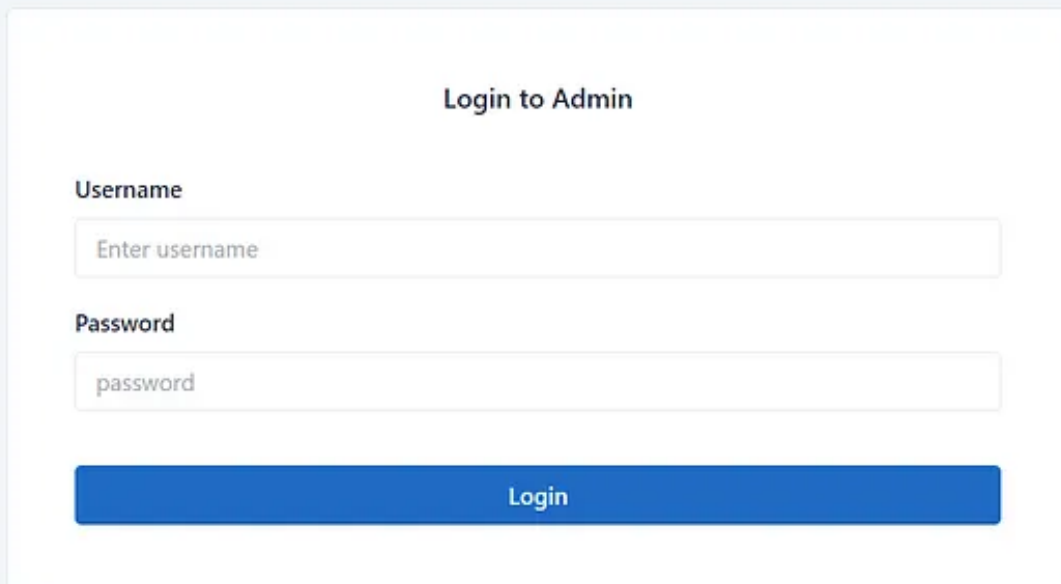
Now run the app

```
uvicorn main:app --reload
```

if everything is okay then go to

```
http://localhost:8000/admin
```

You should see a page that will ask you for username and password



Login to Admin

Username

Password

Login

Once you enter your username and password you'll be able to enter your admin panel.

Conclusion

That was it. If you want more customization then you can add your own template inside the folder “**templates/sqladmin**” with the name “**login.html**” etc to have your own login page. In admin you can do all kinds of operations including edit, delete, add etc.

please follow me for more of such content

[Twitter](#), [Github](#), [Linkedin](#)

Reference:

[Link](#), [Link](#)

Fastapi

Sqladmin

Fastapi Admin

Django Admin



Follow


Written by Shubhendu ghosh

75 Followers

AI/ML software developer at Dolphy

More from Shubhendu ghosh




 Shubhendu ghosh

How to use Hugging Face models in your project.

Hugging Face is a library where you can find machine learning models for your project. It consists of numerous transformers models that are...

May 7, 2023  70  2



 Shubhendu ghosh

Hyperparameter tuning guide

introduction

Mar 18, 2022  61





Shubhendu ghosh

How to use OpenAI Whisper to Transcript Audio

Use Whisper — A huggingface open source model in python to create a web app

Feb 16 🖱️ 8





Shubhendu ghosh

How to write a blog using ChatGPT

Introduction

Dec 30, 2022



130



3



See all from Shubhendu ghosh

Recommended from Medium

```
est % python medium.py
__<module>:5 - I will show how
__<module>:6 - I love how the
__<module>:7 - Different color
__<module>:8 - Easy to get star
__<module>:9 - So many options
__<module>:10 - You see what i
```



Akhilesh Mishra in Level Up Coding

I Ditched Python Built-In Logging For Loguru—You Should Too

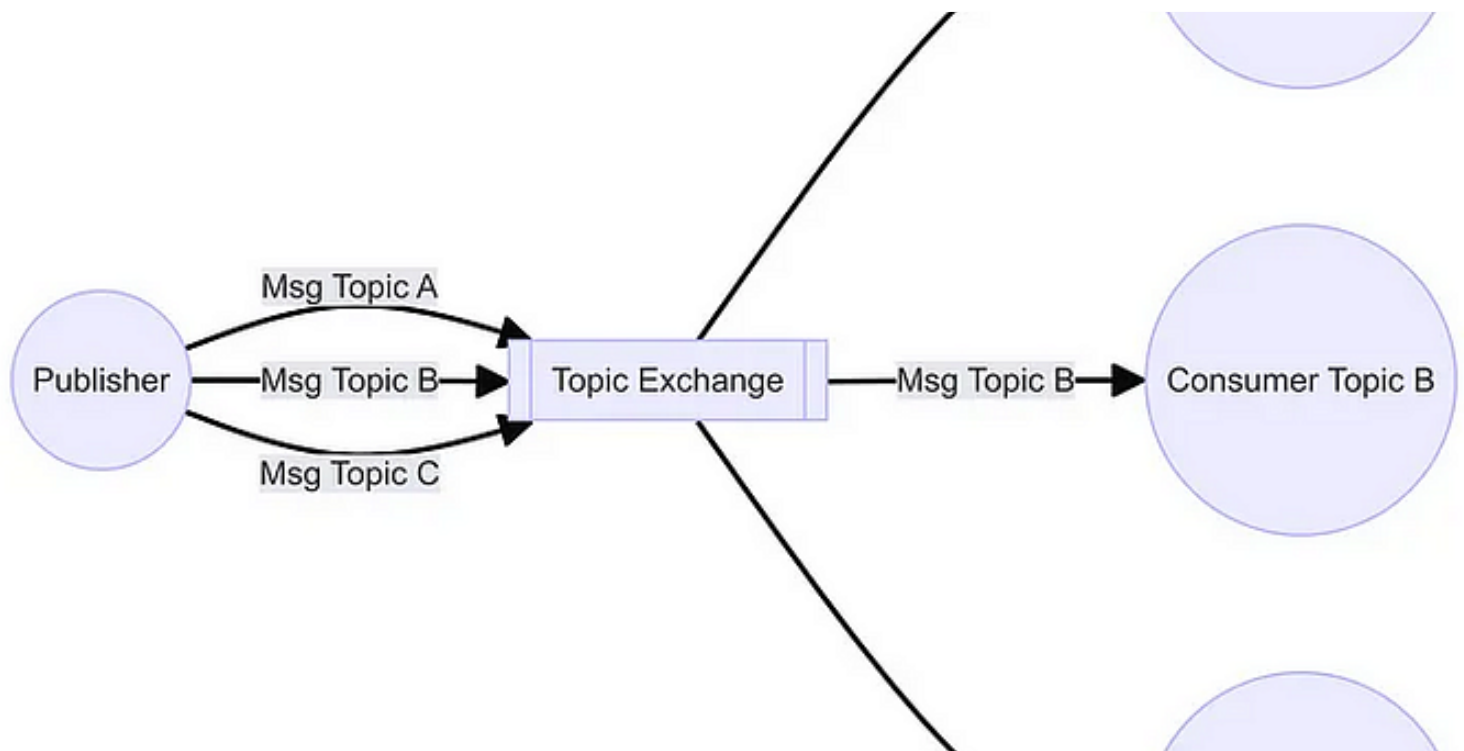
A Complete guide to better logging in Python with loguru



6d ago



265



Marc Nealer

Use aio-pika instead of Celery for Django and FastAPI

I've written a few article on this subject already, but I thought one more to make my point very clear would be a good idea.

Lists



Coding & Development

11 stories · 795 saves



Natural Language Processing

1687 stories · 1254 saves



Lynn G. Kwong in Python in Plain English

Use Ruff to Make Your Python Code More Professional

Learn a new and extremely fast Python linter and code formatter



Aug 31



81

Limit 200MB per file • JPG, PNG, JPEG

Browse files

portrait-3204843_640.jpg 44.3KB


×

Run Script

Before

After



 Tuan Truong

I gathered top 15 ultimate automation python scripts into an open source project

Hey there, tech enthusiasts and productivity geeks! Have you ever found yourself drowning in repetitive tasks, wishing there was an easier...

Jul 16  130  1



 Leo Liu

FastAPI vs. Other Backend Frameworks: The Ultimate Comparison

If you're in the world of backend development, you've likely heard the buzz around FastAPI. It's quickly becoming a go-to framework for...



 Thomas Reid  in AI Advances

Moving from Pandas to DuckDB?

Ten common Pandas operations and their DuckDB equivalents

★ Sep 2 🖱️ 241 💬 1

🔖+ ⋮

See more recommendations