Flask Blueprints Project Structure

Modular Flask app using Blueprints. Organized by feature for medium-sized projects.

#flask #python #web #backend #modular #blueprints

PNG PDF Copy Prompt

Project Directory

```
myproject/
    run.py  Entry point
    config.py  Configuration c…
    requirements.txt
    .env
    .env.example
    .gitignore
  app/  Application pac…
    __init__.py  App factory here
    extensions.py  Flask extension…
    models.py  All SQLAlchemy …
  main/  Main blueprint
    __init__.py  Blueprint regis…
    routes.py  Route handlers
    forms.py
  auth/  Auth blueprint
    __init__.py
    routes.py
    forms.py
    utils.py  Password hashin…
  api/  API blueprint (…
    __init__.py
    routes.py
    schemas.py  Marshmallow sch…
  templates/
    base.html
    main/
      index.html
      about.html
    auth/
      login.html
      register.html
  static/
    css/
    js/
    images/
  tests/
    __init__.py
    conftest.py  Pytest fixtures
    test_main.py
    test_auth.py
```

Why This Structure?

Blueprints break your Flask app into feature modules. Each blueprint (main, auth, api) is self-contained with its own routes, forms, and templates. This structure scales well for teams and lets you add features without touching existing code.

Key Directories

app/__init__.py - Application factory—creates and configures the app

app/extensions.py - Shared extensions: db, migrate, login_manager

app/{blueprint}/ - Each feature is a package with routes.py, forms.py

app/templates/{blueprint}/ - Templates namespaced by blueprint

Blueprint Registration

```
# app/auth/__init__.py
from flask import Blueprint
auth_bp = Blueprint('auth', __name__, template_folder='../

from app.auth import routes  # Import routes after bp crea

# app/__init__.py
from app.auth import auth_bp
app.register_blueprint(auth_bp, url_prefix='/auth')
```

Getting Started

1. `python -m venv venv && source venv/bin/activate`
2. `pip install flask flask-sqlalchemy flask-migrate python-dotenv`
3. `pip freeze > requirements.txt`
4. `cp .env.example .env`
5. `flask db init && flask db migrate -m "Initial"`
6. `flask run`

When To Use This

- Apps with 3+ distinct feature areas
- Projects with multiple developers
- Need to add features without refactoring
- Medium-sized web applications
- Projects expecting 6+ months of development

Trade-offs

More boilerplate - Each blueprint needs __init__.py setup

Circular imports - Must carefully order imports (app factory helps)

Learning curve - Blueprint registration can confuse beginners

Naming Conventions

Blueprints - Lowercase, feature-based (auth, blog, api)

Routes - Functions named view_name or action_resource

Templates - Grouped by blueprint: templates/auth/login.html

Models - PascalCase singular (User, Post, Comment)