DA FLASK AD ANGULAR

AVVENTURA DI 2 STUDENTI NELLA CREAZIONE DI UNA WEB APP CON FLASK

BACARO tech

PIACERE
MI PRESENTO

BACARO tech

Giorgio Basile

Sviluppatore web front-end
Mérieux NutriSciences

MY STACK

@giorgiobasile00

@ghita00

CONNETTIAMOCI!

E ORA
VI PRESENTO LEI

BACARO tech

Romina Alina Danci

System integrator

Avvale

HER STACK

@romina-alina-danci-62ba59248

CONNETTIAMOCI!

CHE COS'È
BACAROTECH

BacaroTech è un collettivo che ha il compito di ricreare quell'atmosfera gioiosa e di gruppo, tipica del bacaro veneziano, nel mondo dell'informatica, attraverso la divulgazione sui social, eventi e workshop

Portiamo avanti questa realtà dal 2023 e nel corso di questo tempo siamo riusciti a mettere a terra una community di appassionati che ammonta a quasi 3000 sviluppatori

CHE COS'È
BACAROTECH

BacaroTech

BacaroTech

Code and Fun

La tua **community di sviluppatori** dove si parla di programmazione a 360°: strutture dati, algoritmi, carriera tech e molto altro!

- Canale Whatsapp
- GitHub
- Instagram
- Youtube
- TikTok
- LinkedIn
- Discord

Condividi questa pagina!

Link della repo di questo LinkTree

Buon codice devs!

MA ADESSO PARLIAMO DI FLASK

PREMESSA
NON MI PIACE PYTHON...

BACARO tech

Preferisco non usare Python, sarà per la sua struttura sintattica oppure per i suoi utilizzi(non l'ho ancora capito), ma lo utilizzo di tanto in tanto, principalmente per scrivere script (che ora sto migrando verso TS) o ho sperimentato la creazione di un bot per Telegram.

PRIMO APPROCCIO
AL LINGUAGGIO SENZA GRAFFE

BACARO tech

La mia prima esperienza con Python risale al corso universitario "Basi di dati" modulo 2.
In quell'occasione, abbiamo applicato le conoscenze acquisite sviluppando un'applicazione che si occupava della gestione di una palestra, utilizzando Flask come framework.

ed è nata questa roba qui

LA MIA TESI
E UN MINIMO COMUNE MULTIPLO

BACARO tech

Da lì mi sono detto “figo, ci faccio la tesi su questo” e, insieme a Romina, abbiamo scelto di lavorare su un progetto congiunto.
Abbiamo deciso di realizzare un gestionale di una pasticceria innovativa e, applicando le tecniche apprese durante il corso di "Ingegneria del Software", abbiamo definito i requisiti e sviluppato il nostro prodotto

ed è nata questa roba qui

FLASK
CHE COS'È?

BACARO tech

"Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications"

FLASK
CHE COS'È? PER DAVVERO

BACARO tech

"Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications"

La mia opinione

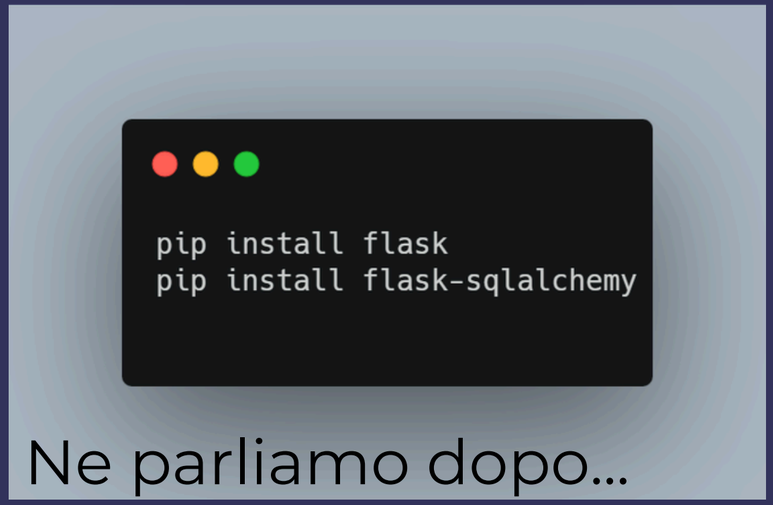
"Ti do il minimo indispensabile per fare le cose ma non eccello in nulla e per questo che le persone preferiscono Django"

FLASK
GESTORE DI DIPENDENZE

BACARO tech

Flask non ha un gestore di dipendenze interno.
Si utilizza pip, il package manager ufficiale di Python.

A differenza di framework come Angular o Spring, non esiste un initializer ufficiale(prossimo progetto Open Sorce?) che generi automaticamente la struttura del progetto con tutte le dipendenze.



FLASK
CARATTERISTICHE

BACARO tech

Flask è un micro-framework web per Python usato per creare:

- Micro-framework → minimalista di base molto comodo per realizzare dei prototipi(POC o Hackathon)
- Routing semplice → attraverso decoratori
- Jinja2 → motore di templating
- Estendibile → data la sua natura minimale
- REST API → utilizzabile anche solo la sua versione “lato server” e basta

SERVER SIDE RENDERING
IL VERO GOAT DEL FRAMEWORK

BACARO tech

Negli ultimi anni molti framework frontend stanno facendo grandi passi avanti (o forse indietro?) su un tema che sembrava superato: il Server-Side Rendering (SSR).

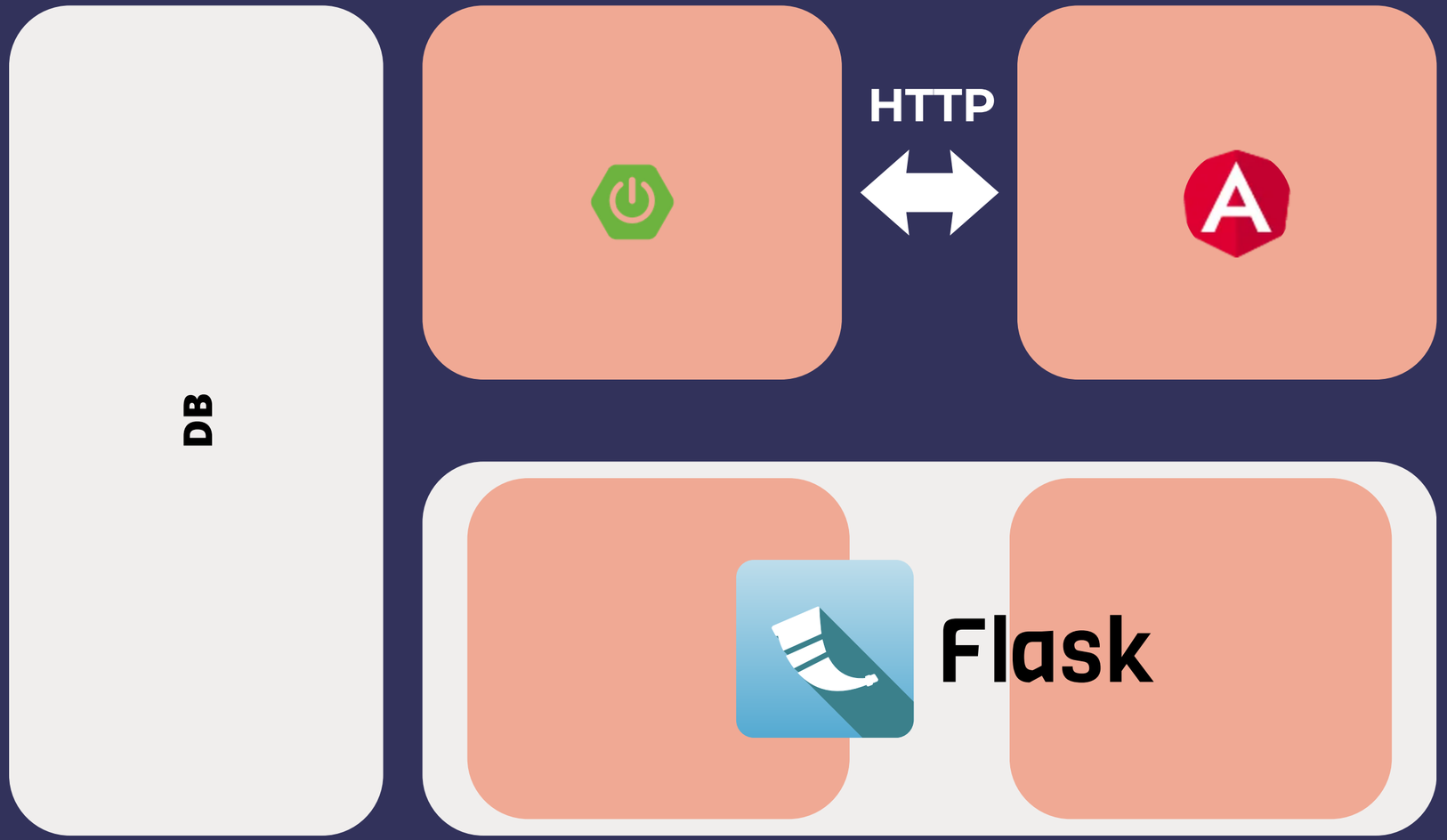
Questo approccio permette di:

- alleggerire il carico sul client — sì, sto guardando te Chrome 👀
- ridurre l’esposizione dei dati, sensibili o meno, che in una SPA pura viaggerebbero tramite chiamate HTTP verso il client
- modelli unici tra la logica di businnes e il fe

... e questo Flask ce l’ha di default(tanta roba)

MODELLO IN COMUNE
MENO CODICE E PIÙ PULITO

BACARO tech



FLASK - FE
TEMPLATE E BINDING

BACARO tech

Un template è un file HTML con variabili e logica dentro, che il server riempie prima di mandarlo al browser, mentre con binding si intende l’operazione per “proiettare” delle variabili della tua logica di businness direttamente nel frontend

🚨 **Punto chiave**
Il binding avviene solo una volta sola lato server prima che la pagina arrivi al browser, quindi, se i dati cambiano serve un refresh o nuova request.

Mentre questo non avviene per i template client-side

FLASK - FE
TEMPLATE E BINDING

BACARO tech

```
<ul>
{% for prodotto in prodotti %}
  <li>{{ prodotto }}</li>
{% endfor %}
</ul>
```

Iterazione

```
{% if user.admin %}
  <p>Sei admin</p>
{% else %}
  <p>Utente normale</p>
{% endif %}
```

Condizione

FLASK - FE
MAIN DI ESEMPIO

BACARO tech

```python
from flask import Flask, request, jsonify

def create_app():
    app = Flask(__name__)

    # Route base
    @app.route("/")
    def home():
        return "Benvenuto nella mia app Flask 🚀"

    # Route con parametro dinamico
    @app.route("/user/<string:name>", methods=["GET", "POST"])
    def user_profile(name):
        return f"Ciao {name}!"

if __name__ == "__main__":
    app = create_app()
    app.run(debug=True)
```

Il file principale di un’app Flask è il punto di avvio dell’applicazione.

Qui vengono:

- Creata l’istanza dell’app
- Definite le route
- Configurate eventuali impostazioni
- Avviato il server di sviluppo

FLASK - FE
BLUEPRINT

BACARO tech

I Blueprint in Flask permettono di organizzare un’applicazione in moduli riutilizzabili, separando le diverse funzionalità in componenti indipendenti.

🎯 Perché usarli?

- Suddividere il codice in moduli logici (es. auth, admin, API)
- Migliorare manutenibilità e scalabilità
- Favorire il riuso del codice
- Separare rotte

FLASK - FE
BLUEPRINT

BACARO tech

```python
#blueprint
from Profile import profile
from Ecommerce import ecommerce
from Blog import blog
from Documenti import documenti
from Magazzino import magazzino
from Ordini import ordini
from Personale import personale
from Ricettario import ricettario
from Produzione import produzione

#registrazione blueprint
app.register_blueprint(profile, url_prefix = "")
app.register_blueprint(ecommerce, url_prefix = "")
app.register_blueprint(blog, url_prefix = "")
app.register_blueprint(personale, url_prefix = "")
app.register_blueprint(ordini, url_prefix = "")
app.register_blueprint(ricettario, url_prefix = "")
app.register_blueprint(documenti, url_prefix = "")
app.register_blueprint(magazzino, url_prefix = "")
app.register_blueprint(produzione, url_prefix = "")
```

```python
profile = Blueprint('profile', __name__)

@profile.route('/login', methods=['GET', 'POST'])
def login():
  ...business logic...
```

COMPONENTE SPECIFICO

FLASK - FE
ROUTING

Il routing è il meccanismo che collega un URL a una funzione Python (view function) che genera la risposta HTTP.

Supporta i parametri dinamici e multi method HTTP

```python
@app.route("/user/<username>", methods=["GET", "POST"])
def profile(username):
    return f"Profilo di {username}"
```

FLASK - BE
SESSIONI

BACARO tech

```python
from flask import Flask, session

app = Flask(__name__)
app.secret_key = "super_secret_key"

@app.route("/login")
def login():
    session["user"] = "Giorgio"
    return "Utente salvato in sessione"

@app.route("/profile")
def profile():
    user = session.get("user")
    return f"Utente loggato: {user}"
```

Le sessioni in Flask permettono di salvare informazioni relative a un utente tra una richiesta HTTP e l'altra.

Poiché HTTP è stateless, Flask utilizza le sessioni per mantenere dati come info temporanee

In Flask le sessioni:

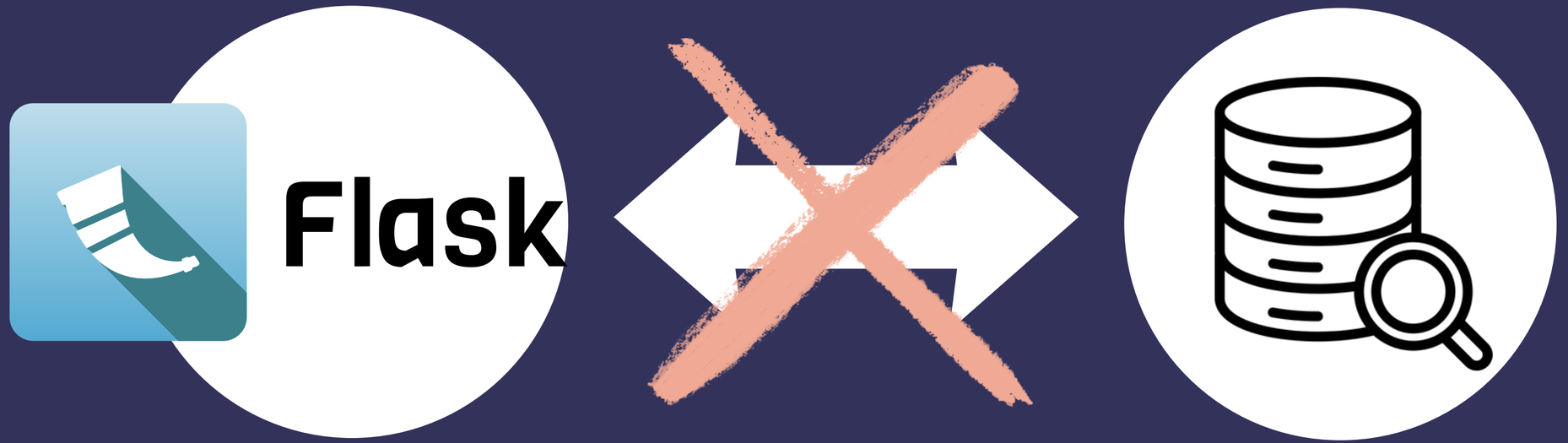
- Sono basate su cookie firmati
- I dati sono salvati lato client
- Sono protetti tramite una SECRET_KEY

FLASK - BE
CONNESSIONI A DB

BACARO tech

Flask è un micro-framework quindi non include di default un sistema di connessione al database.

Ma significa che dobbiamo integrare noi un ORM o un driver DB, ed un esempio che viene spesso usato è Flask-SQLAlchemy



SQL ALCHEMY
CONNESSIONI A DB

BACARO tech

SQLAlchemy è un toolkit SQL + ORM per Python.

Permette di lavorare con database relazionali in modo: astratto (tramite ORM) o esplicito (tramite SQL Core) Indipendente dal database
Supporta: SQLite, PostgreSQL, MySQL, Oracle, SQL Server.

SQL ALCHEMY
CONNESSIONI A DB

BACARO tech

SQLAlchemy è un toolkit SQL + ORM per Python.

Permette di lavorare con database relazionali in modo: astratto (tramite ORM) o esplicito (tramite SQL Core) Indipendente dal database
Supporta: SQLite, PostgreSQL, MySQL, Oracle, SQL Server.

SQLAlchemy Core

- Costruzione esplicita di query SQL
- Più vicino al SQL tradizionale
- Maggiore controllo

ORM (Object Relational Mapper)

- Tabelle → Classi Python
- Righe → Oggetti
- Query → Metodi Python

SQL ALCHEMY
CONNESSIONI A DB

BACARO tech

```python
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config["SQLALCHEMY_DATABASE_URI"] = "sqlite:///app.db"

db = SQLAlchemy(app)

#----

class User(db.Model):
    __tablename__ = "users"

    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True)
```

Esempio di:

- Definizione della connesione tramite app.config
- Definire modelli come classi Python che estrendo db.Model

Eventualmente si possono gestire relazioni tra tabelle 1-1, 1-n, n-n

SQL ALCHEMY - CORE
CONNESSIONI A DB

BACARO tech

```python
from sqlalchemy import select, func, desc

stmt = (
    select(
        fatture_acquisto.c.Status,
        fatture_acquisto.c.Id,
        fatture_acquisto.c.Data,
        ditta_fornitrice.c.Mail,
        ditta_fornitrice.c.NomeDitta,
        func.sum(
            contenuto_acquisto.c.Quantità *
            (
                merce.c.PrezzoUnitario +
                ((merce.c.IVA / 100) * merce.c.PrezzoUnitario)
            )
        ).label("Totale")
    )
    .select_from(
        fatture_acquisto
        .join(ditta_fornitrice, ditta_fornitrice.c.PartitaIVA == fatture_acquisto.c.Id_Fornitore)
        .join(contenuto_acquisto, contenuto_acquisto.c.Id_FatturaAcquisto == fatture_acquisto.c.Id)
        .join(merce, merce.c.Id == contenuto_acquisto.c.Id_Merce)
    )
    .group_by(
        fatture_acquisto.c.Id,
        fatture_acquisto.c.Data,
        ditta_fornitrice.c.NomeDitta,
        ditta_fornitrice.c.Mail,
        fatture_acquisto.c.Status
    )
    .order_by(desc(fatture_acquisto.c.Data))
)

result = session.execute(stmt).all()
```

Esempio di una query eseguita tramite Sql Alchemy Core

Mamma mia quante righe😩

Tu dopo aver scritto questo

SQL ALCHEMY - ORM
CONNESSIONI A DB

BACARO tech

Esempio della stessa query eseguita tramite Sql Alchemy ORM che, per essere usato, richiede di aver definito correttamente i model
Già molto meglio😁

```
fatturaAcq = session.query(FattureAcquisto.Status, FattureAcquisto.Id, FattureAcquisto.Data,
DittaFornitrice.Mail,
                           DittaFornitrice.NomeDitta, func.sum(ContenutoAcquisto.Quantità *
(Merce.PrezzoUnitario + ((Merce.IVA / 100) * Merce.PrezzoUnitario))).label('Totale')). \
        join(DittaFornitrice, DittaFornitrice.PartitaIVA == FattureAcquisto.Id_Fornitore). \
        join(ContenutoAcquisto, ContenutoAcquisto.Id_FatturaAcquisto == FattureAcquisto.Id). \
        join(Merce, Merce.Id == ContenutoAcquisto.Id_Merce). \
        group_by(FattureAcquisto.Id, FattureAcquisto.Data, DittaFornitrice.NomeDitta,
DittaFornitrice.Mail). \
        order_by(desc(FattureAcquisto.Data)). \
        all()
```

FLASK
IN CONCLUSIONE

BACARO tech

In conclusione possiamo dire che Flask è ottimo per costruire web server e API leggere, ma non è un framework full-stack.

Punti di forza:

- Leggero, facile e flessibile
- Ampio ecosistema di estensioni

Limitazioni(per alcuni):

- Non include nessuna gestione nativa
- Sei tu a scegliere librerie ed estensioni per funzionalità full-stack

FLASK
ALTERNATIVE

FastAPI

BACARO tech

Un’alternativa anche più valida di Flask è FastAPI, ovvero un framework Python moderno per la creazione di API, che può essere utilizzato come alternativa a Flask.

Caratteristiche principali:

- Basato su Starlette e ASGI per prestazioni elevate
- Supporta programmazione asincrona (async/await)
- Validazione e parsing automatico dei dati con Pydantic
- Ideale per API REST e microservizi
- Generazione automatica della documentazione (Swagger / ReDoc)

MONOLITE VS MICROSERVICE

SI STAVA MEGLIO QUANDO SI STAVA PEGGIO(?)

BACARO tech

Quando si sviluppa un’applicazione, l’architettura scelta influisce su scalabilità, manutenibilità e tempi di sviluppo. Le due principali filosofie sono il monolite e i microservizi.

Da quelle che sono state le evoluzioni del web sembrava che la strada dei microservizi fosse la definitiva ma come abbiamo visto abbiamo già fatto dei passi indietro, quindi quale sarà il futuro del web secondo voi?

BACARO tech

VI RINGRAZIA TUTTI PER AVER PARTECIPATO!

ASPETTIAMO VOSTRE DOMANDE

BACARO TECH

SEGUITECI!

BACARO tech

BacaroTech

BacaroTech

Code and Fun

La tua **community di sviluppatori** dove si parla di programmazione a 360°: strutture dati, algoritmi, carriera tech e molto altro!

- Canale Whatsapp
- GitHub
- Instagram
- Youtube
- TikTok
- LinkedIn
- Discord

Condividi questa pagina!

Link della repo di questo LinkTree

Buon codice devs!