Implementare CORS da zero in Python con Flask

CORS (Cross-Origin Resource Sharing) è un meccanismo che consente a un'applicazione web in esecuzione su un dominio di accedere alle risorse di un altro dominio. In Flask, si può usare l'estensione `flask-cors`, ma in questo articolo vedremo come implementarlo manualmente, da zero.

Cos'è CORS?

Quando un browser effettua una richiesta da un'origine diversa da quella del server, invia automaticamente degli header come `Origin` e, in alcuni casi, effettua una richiesta **preflight** (con metodo `OPTIONS`). Il server deve rispondere con header CORS specifici per autorizzare l'accesso.

Obiettivo

Permettere richieste da un'origine specifica (es. `http://localhost:3000`) rispondendo con gli header corretti.

1. Impostare Flask

```
from flask import Flask, request, jsonify

app = Flask(__name__)
```

2. Middleware CORS personalizzato

Creiamo una funzione da eseguire dopo ogni richiesta per aggiungere gli header necessari:

```
@app.after_request
def add_cors_headers(response):
    response.headers['Access-Control-Allow-Origin'] = 'http://localhost:3000'
    response.headers['Access-Control-Allow-Methods'] = 'GET, POST, PUT,
DELETE, OPTIONS'
    response.headers['Access-Control-Allow-Headers'] = 'Content-Type,
Authorization'
    return response
```

3. Gestire le richieste OPTIONS (preflight)

Flask, di default, non risponde automaticamente con i giusti header alle richieste `OPTIONS`. Li gestiamo manualmente:

```
@app.route('/api/data', methods=['GET', 'POST', 'OPTIONS'])
def handle_data():
    if request.method == 'OPTIONS':
        response = app.make_response('')
        response.headers['Access-Control-Allow-Origin'] =
'http://localhost:3000'
        response.headers['Access-Control-Allow-Methods'] = 'GET, POST, PUT,
DELETE, OPTIONS'
        response.headers['Access-Control-Allow-Headers'] = 'Content-Type,
Authorization'
        return response, 204

    data = {'message': 'CORS abilitato'}
    return jsonify(data)
```

4. Avvio dell'app

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

Conclusione

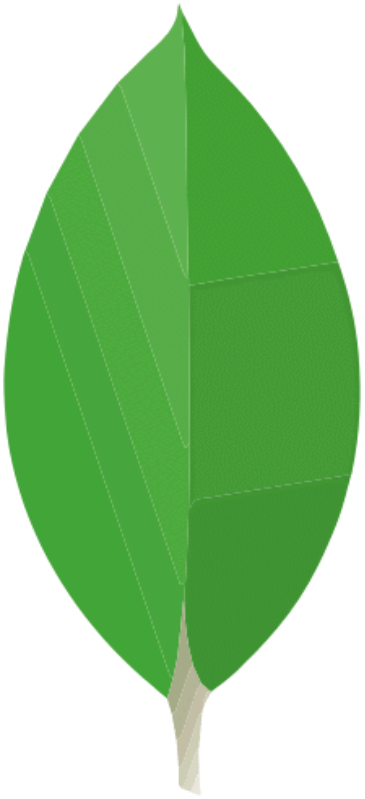
Con questa semplice implementazione, hai pieno controllo su come vengono gestite le policy CORS nella tua app Flask. Questo approccio è utile quando vuoi evitare dipendenze esterne o comprendere a fondo il funzionamento di CORS.

Applicazioni Correlate

-

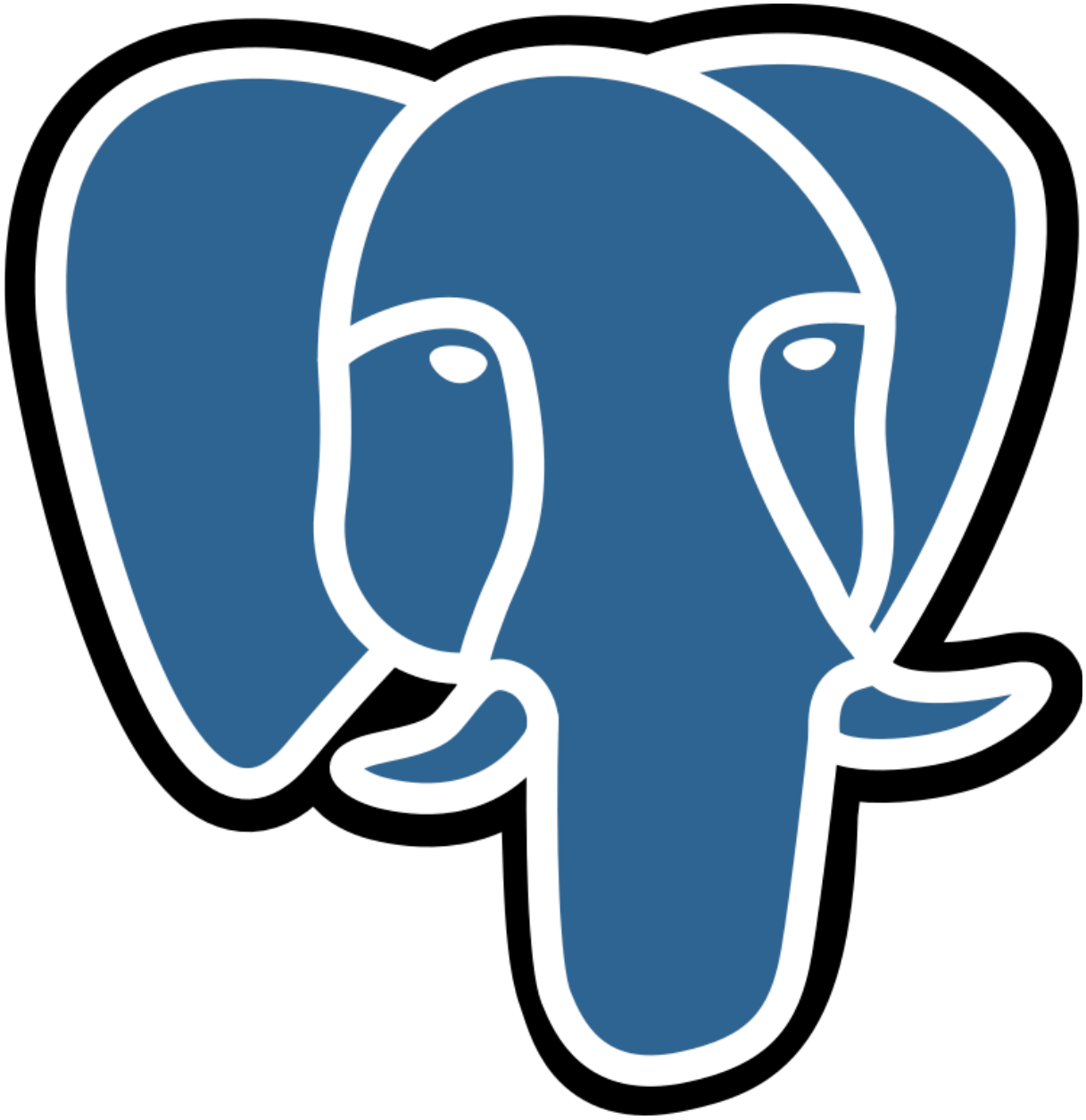
Python Placeholder Image

Applicazione sviluppata in Python con Flask per la creazione di immagini segnaposto.
DockerDocker ComposePythonFlaskJavaScript



Python MongoDB App

Applicazione basata su MongoDB e sviluppata in Python e Flask.
DockerDocker ComposePythonFlaskMongoDB



Python PostgreSQL App

Applicazione basata su PostgreSQL con Python e Flask.
DockerDocker ComposePythonFlask