Politecnico di Torino

# Applicazioni Web

# Applicazioni Web

- Introduzione
- Sviluppo
- Interazione con il DBMS

# Introduzione

Applicazione web

# Che cos’è un’applicazione web?

Un’applicazione ospitata da un server remoto e utilizzata dagli utenti via internet attraverso un browser (e.g. Chrome, Safari)

**Benefici**

- L’utente non necessita di installare o aggiornare l’app
- L’utente può accedere al servizio da diversi dispositivi e browser
- Ridotti problemi di compatibilità
- Facilità di *deployment* a manutenzione

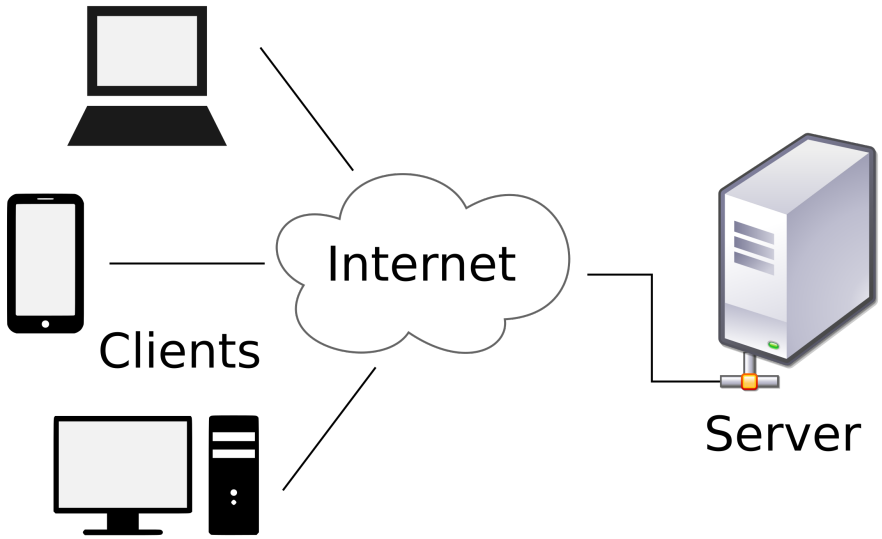
# Architettura Client-Server

Architettura distribuita composta da un Client e un Server che comunicano attraverso uno specifico protocollo

**Client**: Effettua la richiesta di un servizio o di una risorsa esposta dal *Server*

**Server**: Fornisce il servizio, ricevendo e processando la richiesta del *Client*

**Protocollo**: Regole e procedure standardizzate che definiscono la comunicazione tra *Client* e *Server* (e.g. HTTP, FTP, SMTP…)



# Protocollo HTTP

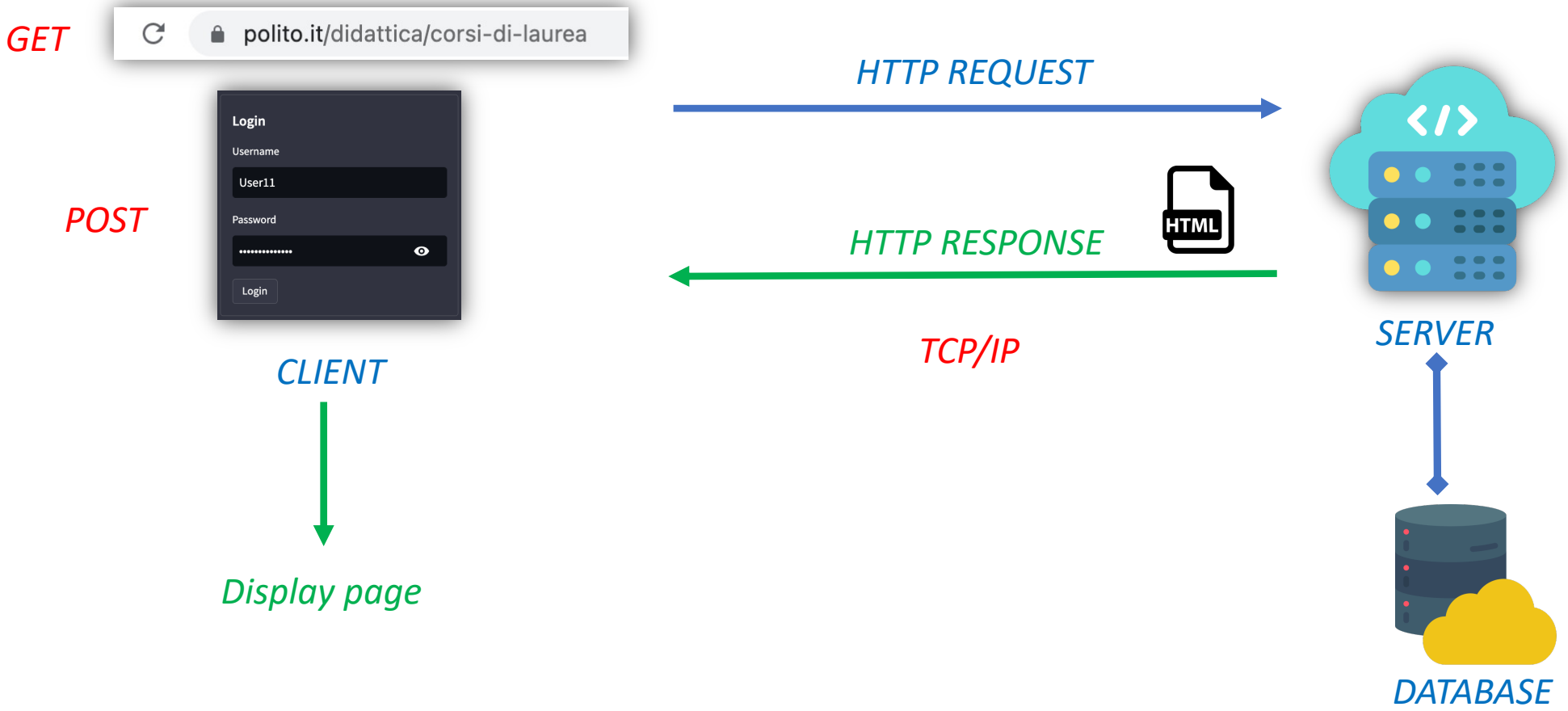

- **HyperText Transfer Protocol** (HTTP) è il più diffuso per la comunicazione tra web-server e web-client, basato sul paradigma *request-response*
- Utilizzato dal Client per effettuare la richiesta al Server
- La richiesta è composta (non esclusivamente) da un metodo e dall'URI, percorso che identifica univocamente la risorsa

| GET | POST | PUT | DELETE |
|---|---|---|---|
| Recuperare dati | Inserire dati | Aggiornare dati o inserirli se non esistono | Eliminare la risorsa specifica |

*Principali metodi HTTP*

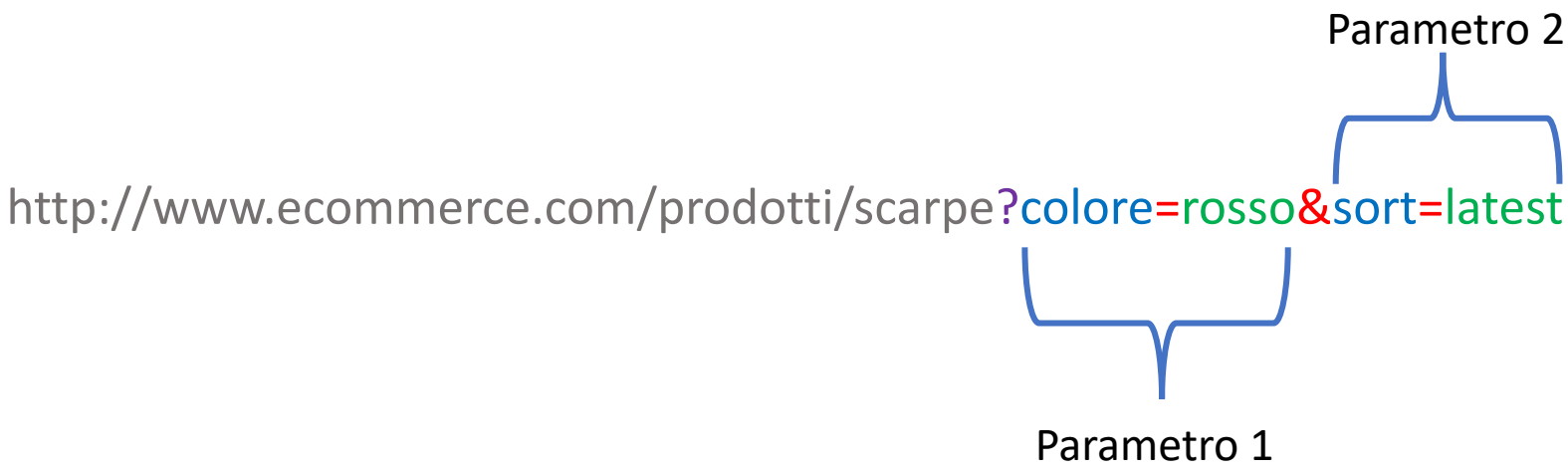
# Protocollo HTTP

- Quando il Server riceve la richiesta, la processa ed eventualmente restituisce la risorsa desiderata (e.g. pagina HTML)



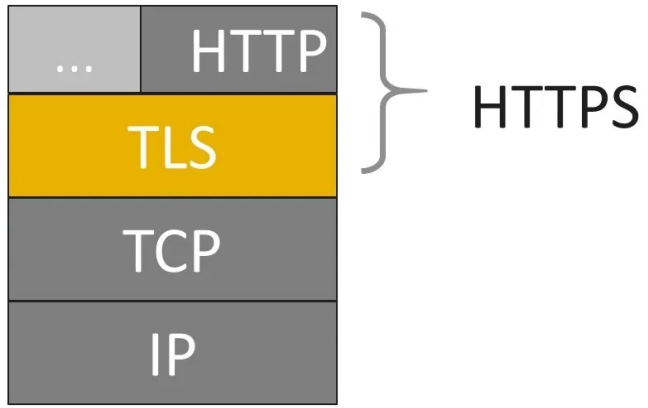
# Protocollo HTTP – Parametri URL

- I parametri URL (noti anche come *query string*) permettono di passare campi addizionali come input della pagina web, per filtrare, completare le interrogazioni, ecc..
- I parametri iniziano dopo il carattere '?' e sono separati da '&'
- Sono strutturati nella forma *key = value*



# Protocollo HTTP – HTTPS

- HyperText Transfer Protocol Secure: protocollo per la comunicazione sicura e crittografata tra client e server
- Usa il protocollo TLS/SSL per criptare le informazioni sulla base di un certificato che autentica il web server
- Protegge dati sensibili da attacchi hacker (*man in the middle, eavesdropping…)*
- Nuovo standard, per cui i browser segnalano come non sicuri i siti che non utilizzano HTTPS



# Protocollo HTTP – Status Code

- Oltre ad eseguire il task e fornire la risorsa richiesta, il Server restituisce uno Status Code (codice numerico) che indica l'esito della richiesta effettuata
- Esistono diversi tipi di messaggi, a loro volta raggruppati in 5 classi distinte:

| 1xx | 2xx | 3xx | 4xx | 5xx |
|---|---|---|---|---|
| Informativo | Successo | Reindirizzamento | Errore lato Client | Errore lato Server |

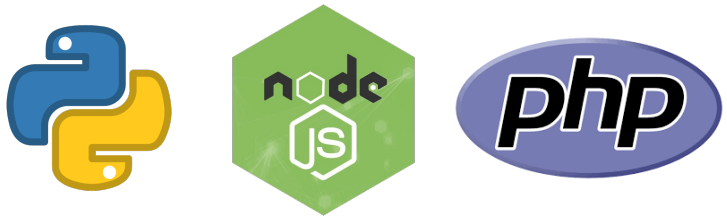
- *Esempio. HTTP Status Code: 404 – "Not Found"*

# Sviluppo

Applicazioni web

# Sviluppo applicazioni web

- La parte di sviluppo di un'applicazione web si divide in diverse componenti, ognuna con i suoi linguaggi e tool specifici

| ***FRONTEND*** | ***BACKEND*** | ***DATABASE*** |
|---|---|---|
| *Parte visibile dell'applicazione con cui l'utente interagisce direttamente.* | *Si occupa del funzionamento del server, dell'elaborazione delle richieste e dell'accesso al database.* | *Si occupa dello storage dei dati, preferenze e informazioni.* |
|   |   |   |

# Sviluppo applicazioni web

- Sia per frontend che per backend esistono diversi *framework* di sviluppo, librerie software che forniscono una base comune per lo sviluppo di un'applicazione in modo più efficiente e organizzato

***FRONTEND***

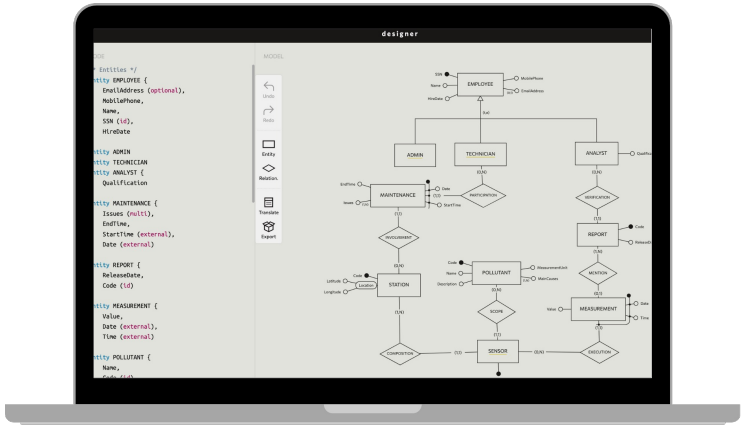
**REACT**

**ANGULAR**

**VUE JS**

**FLUTTER**

Esempio: designer è stato sviluppato in Vue JS




**designER**

***BACKEND***

Express JS

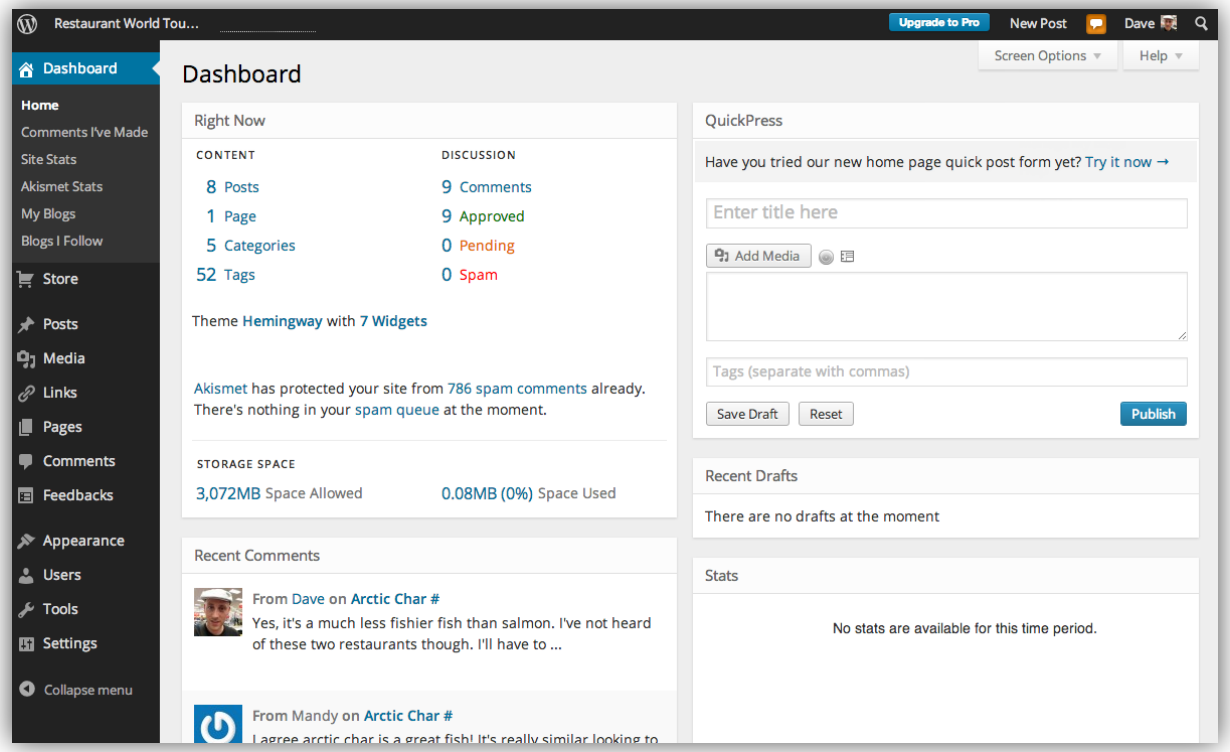
python django

RAILS

php

# Contente Management System (CMS)

- Programma per la gestione e creazione di siti web
- Nessun bisogno di scrivere codice o avere particolari competenze informatiche
- Si possono creare siti, blog, eCommerce, forum, ecc…
- Popolari: Wordpress, Wix, Drupal



# Il linguaggio HTML

- HTML: HyperText Markup Language
- Standard “de facto”
- Scopo: fornire una descrizione strutturata di un documento ipertestuale indipendente dai programmi
- Utilizzato per creare pagine web, definendone la struttura, il contenuto e il layout

**Linguaggio Markup <> Linguaggio di programmazione**

# Il linguaggio HTML - Tag

- HTML consente di annotare un testo per contrassegnare le parti che lo compongono attraverso i *tag*
- I tag sono espressioni sempre racchiuse tra i simboli di minore (<) e maggiore (>)
- Di solito porzioni di testo sono delimitate da coppie di tag (es: <h1>Titolo</h1>)

| TAG | UTILIZZO |
|---|---|
| **<head>** | Informazioni relative al documento HTML |
| **<body>** | Contenuto della pagina web |
| **<h1>**, **<h2>**, **<h3>**, **<h4>** […] | Titoli di sezione di diverso livello |
| **<p>** | Paragrafo |
| **<table>** | Tabella |

# Il linguaggio HTML - Attributi

- Tramite gli attributi è possibile caratterizzare meglio un tag
- Gli attributi sono costituiti da una variabile a cui viene assegnato un valore particolare

Esempio:

```
<!DOCTYPE html>
<html lang="it">
<head>
  <meta charset="UTF-8">
  <title>Basi di Dati</title>
</head>
<body style="background-color:powderblue;">
  <h1 style="color:red;">Basi di Dati: Sviluppo Web</h1>
  <p style="font-family:courier">Questa è una pagina web HTML di esempio. </p>

  <h2>Che cos'è HTML?</h2>
  <p>HTML è l'acronimo di HyperText Markup Language, ovvero linguaggio di markup per ipertesti.</p>

  <h2>Come imparare di più su HTML?</h2>
  <p>Esistono molte risorse online:</p>
  <ul>
    <li><a href="https://developer.mozilla.org/it/docs/Web/HTML">Documentazione su Mozilla Developer</a></li>
  </ul>

  <p style="font-size:160%">Buono studio!</p>
</body>
</html>
```

# Il linguaggio HTML - Panoramica


- HTML consente principalmente di creare pagine web statiche
- Permette di includere: immagini, audio, video, tabelle, form, collegamenti ipertestuali…
- Colori, font, sfondi possono essere gestiti anche da altri linguaggi come il ***CSS***
- Per creare pagine web dinamiche (create “al volo” in risposta all’input dell’utente) si utilizzano linguaggi come JS
- In alcuni casi, la pagina web è generata dinamicamente lato server che restituisce al client la pagina statica

# I Fogli di Stile

Introdotti con HTML 4 per:

- potenziare la descrizione degli aspetti di presentazione/stile
- permettere la separazione tra presentazione e contenuto

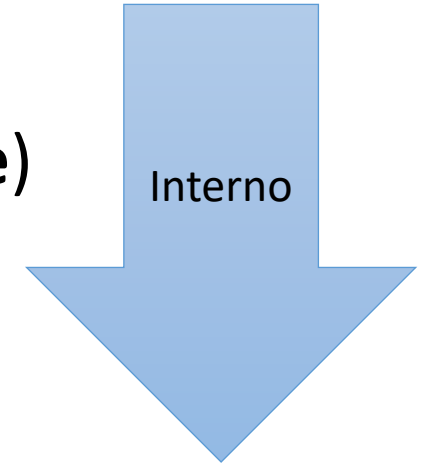
Le specifiche di Stile possono essere indicate sia nel documento HTML che in file separati

- **Specifiche di Stile in-line**: lo stile è specificato direttamente all'interno dell'elemento HTML
- **Specifiche di Stile Interne**: lo stile è specificato direttamente all'interno del documento HTML
- **Specifiche di Stile Esterne**: lo stile è specificato al di fuori del documento HTML in fogli di stile separati

# I Fogli di Stile - Gerarchie

La gerarchia delle regole CSS indica che le regole sono applicate nel seguente ordine:

1. Attraverso un file esterno (**external style**)
2. Con tag <style> interni al documento HTML (**internal style**)
3. Specificando elementi di stile in un qualsiasi tag del documento tramite l'attributo style (**in-line style**)




*In caso di conflitto, si applica quella più interna!*

# I Fogli di Stile- Bootstrap

- Framework front-end open source che mette a disposizione una collezione di classi CSS e funzioni JS pronte all'uso

  *Layout a griglia, tabelle, form, tipografia, pannelli e altro*

- Segue le proprietà del design responsivo

  *Utilizzare CSS e HTML per ridimensionare, nascondere, contrarre, allargare o spostare il contenuto della pagina web*

- Rende veloce e personalizzabile lo sviluppo web

https://getbootstrap.com/docs/4.0/getting-started/introduction/

# Python Backend Framework

Per lo sviluppo web in Python esistono diversi framework e micro-framework

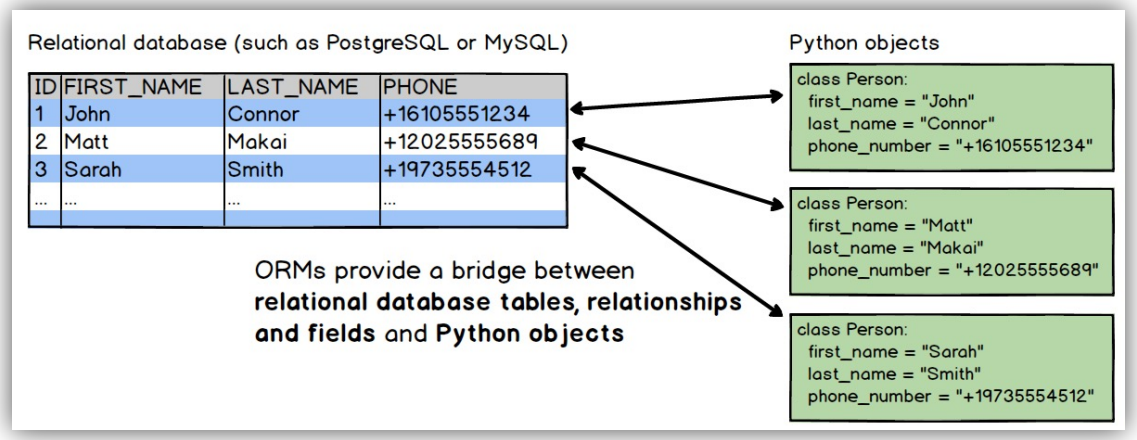

- Micro-framework
- Jinja2 template
- ORM gestito da altri pacchetti
- nessuna interfaccia admin
- nessun sistema di autenticazione


- Built-in tool per encoding, sessioni, caching, autenticazione, contenuto statico
- ORM gestito da altri pacchetti


- Applicazioni complesse
- ORM
- Model-Template-View
- Sistema di autenticazione built-in
- Interfaccia admin inclusa

# Object-Relational Mapping (ORM)

- Integrare i sistemi RDBMS all'interno del paradigma della programmazione orientata agli oggetti (**OOP**)
- Permette un'astrazione ad alto livello per semplificare lo sviluppo invece di usare direttamente l'SQL
- **Portabilità** tra un DBMS e un altro
- Rischio di riduzione di performance




https://www.fullstackpython.com/object-relational-mappers-orms.html

# Flask

- Micro-framework Python per la creazione di applicazioni web
- Comprende un web server accessibile in locale
  - *localhost (http://127.0.0.1, http://localhost)*
  - *porta default 5000*
- Approccio *Pythonic*, flessibile e facile da imparare

```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/hello")
def hello():
    return render_template("hello.html",name="Pedro")

app.run(port=8080,debug=True)
```

app.py

```html
<html>
<head>
    <title>Hello Page </title>
</head>
<body>
    {% if name}
        <h1> Welcome {{name}} </h1>
</body>
</html>
```

templates/hello.html

# Django

- Django è un framework web Python di alto livello che favorisce uno sviluppo rapido e un design pulito e pragmatico[1]
- Viene considerato con “batterie incluse” in quanto è un framework completo di tutti gli elementi necessari per gli sviluppatori
- MVT design pattern (Model View Template)

| Models | Views | Templates | URLs |
|---|---|---|---|
| Definiscono la struttura dei dati con i meccanismi di interrogazione del database | Funzioni che ricevono una richiesta HTTP e restituiscono la risposta HTTP | Descrivono il layout con cui i risultati devono essere rappresentati (e.g. file HTML) | Mappatura per indirizzare la richiesta HTTP alla view corretta |
| models.py | views.py | /templates | urls.py |

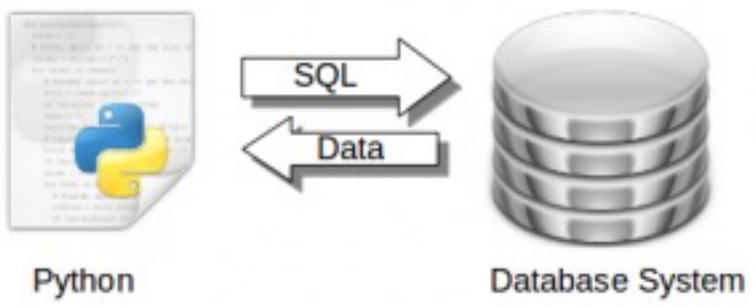
[1] https://www.djangoproject.com

# Interazione con il DBMS

Applicazioni web

# Panoramica

- Un applicazione web necessita di interfacciarsi con un database per effettuare le interrogazioni
- Python ha moduli per interfacciarsi con i principali DB: MySQL, PostgreSQL, Oracle, MariaDB…
- Esiste una strutturazione comune dell'interazione con il DBMS:
  1. Apertura della connessione con il DBMS
  2. Esecuzione di istruzioni SQL
  3. Chiusura della connessione



# SQLAlchemy

- SQLAlchemy è una libreria Python che permette di interfacciarsi con un database in modo efficiente
- Offre la flessibilità e l'efficacia di SQL all'interno della propria applicazione Python
- Funzionalità supportate:
  1. Connessione al DB
  2. Esecuzione immediata di query SQL
  3. Acquisizione e lettura di dati
  4. Query multiple e transazioni

# SQLAlchemy – Apertura connessione

- Starting point delle applicazioni che utilizzano SQLAlchemy, permette di specificare i dettagli della connessione
- Richiede cinque parametri:
  1. **dialect**: nome del linguaggio che verrà utilizzato per la connessione
  2. **username**: nome dell'utente nel db
  3. **password**: password dell'utente
  4. **host**: nome della macchina che ospita il DBMS
  5. **dbname**: nome del DB

```
1   from sqlalchemy import create_engine
2
3   dialect = "mysql"
4   username="root"
5   password=""
6   host="127.0.0.1"
7   dbname="Opere"
8
9   engine=create_engine(f"{dialect}://{username}:{password}@{host}/{dbname}")
10
11  conn=engine.connect()
12
```

- Chiamata alla funzione ***create_engine()***
- Connessione al DB con la funzione ***connect()***

# SQLAlchemy – Interrogazione SQL

- Esecuzione immediata dell'istruzione: il server compila ed esegue immediatamente l'istruzione SQL ricevuta
- Chiamata alla funzione ***execute()***
- Richiede come parametro la query da eseguire, in formato stringa
- In caso di successo restituisce il risultato della query, in caso di insuccesso solleva un'eccezione
- Il risultato viene memorizzato con una variabile di tipo "***cursor***"

```
12
13    myquery="SELECT autore.cognome, opera.nome\
14             FROM autore, opera\
15             WHERE autore.coda=opera.autore"
16
17    result=conn.execute(myquery)
```

# SQLAlchemy – Transazioni

- Le connessioni avvengono implicitamente in modalità ***auto-commit***

Dopo l'esecuzione con successo di ogni istruzione SQL, è eseguito automaticamente il commit

- Bisogna impostare un ***commit non automatico*** per eseguirlo dopo una sequenza di istruzioni SQL

Si esegue un solo commit alla fine dell'esecuzione di tutte le istruzioni

# SQLAlchemy – Transazioni

- Chiamata alla funzione ***begin()***
- Quando invocata SQLAlchemy inizializza una transazione e disaliblita l'autocommit
- In caso di successo restituisce una transazione attiva, altrimenti solleva un'**eccezione**

```
18
19    #Initialize a new transaction
20    myTransaction=conn.begin()
21
```

# SQLAlchemy – Transazioni

- Se si disabilita l'autocommit le operazioni di ***commit*** e ***rollback*** devono essere richieste esplicitamente

***commit()***

- Esegue il commit della transazione corrente
- In caso di insuccesso solleva un eccezione

```
22    #Commit the operations
23    myTransaction.commit()
24
```

***rollback()***

- Esegue il rollback della transazione corrente
- In caso di insuccesso solleva un eccezione

```
25    #Rollback the operations
26    myTransaction.rollback()
27
```

# SQLAlchemy – Transazioni

- Se si disabilita l'autocommit le operazioni di ***commit*** e ***rollback*** devono essere richieste esplicitamente

- Utilizzando il costrutto ***with*** SQLAlchemy gestisce automaticamente il *commit* o il *rollback*
  - *Esegue il commit in caso di successo*
  - *In caso di insuccesso esegue il rollback solleva un eccezione*

```
28    #Initialize a transaction and Commit or Rollback
29    with conn.begin():
30        #... SQL and SQLAlchemy code ...
31
```

# SQLAlchemy – Chiusura connessione

- Deve essere eseguita quando non è più necessario interagire con il DBMS
- Chiude il collegamento con il DBMS e rilascia le relative risorse
- Chiamata alla funzione ***close()***

```
18
19    #Close the DB connection
20    conn.close()
```