Politecnico
di Torino

DBMG

Web Applications

Web Applications

- Introduction
- Development
- Interaction with the DBMS

Introduction

Web Applications

What is a web application?

An application hosted by a remote server and used by users via the internet through a browser (e.g., Chrome, Safari)

Pros

- User does not need to install or update the app
- The user can access the service from different devices and browsers
- Reduced compatibility issues
- Easy *deployment* and maintenance

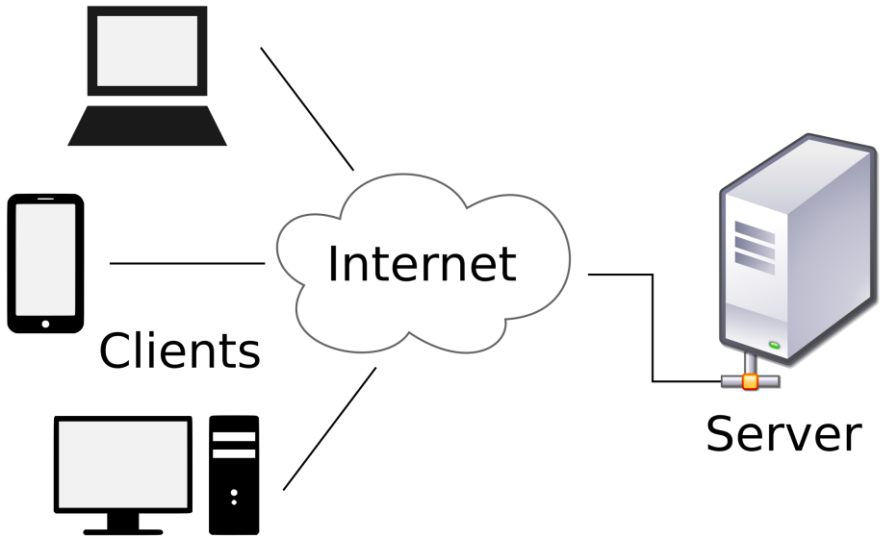
Client-Server Architecture

Distributed architecture composed of a Client and a Server that communicate through a specific protocol

Client: requests a service or resource exposed by the *Server*

Server: provides the service, receiving and processing the *Client*'s request

Protocol: standardized rules and procedures that define the communication between *Client* and *Server* (e.g., HTTP, FTP, SMTP...)



HTTP Protocol

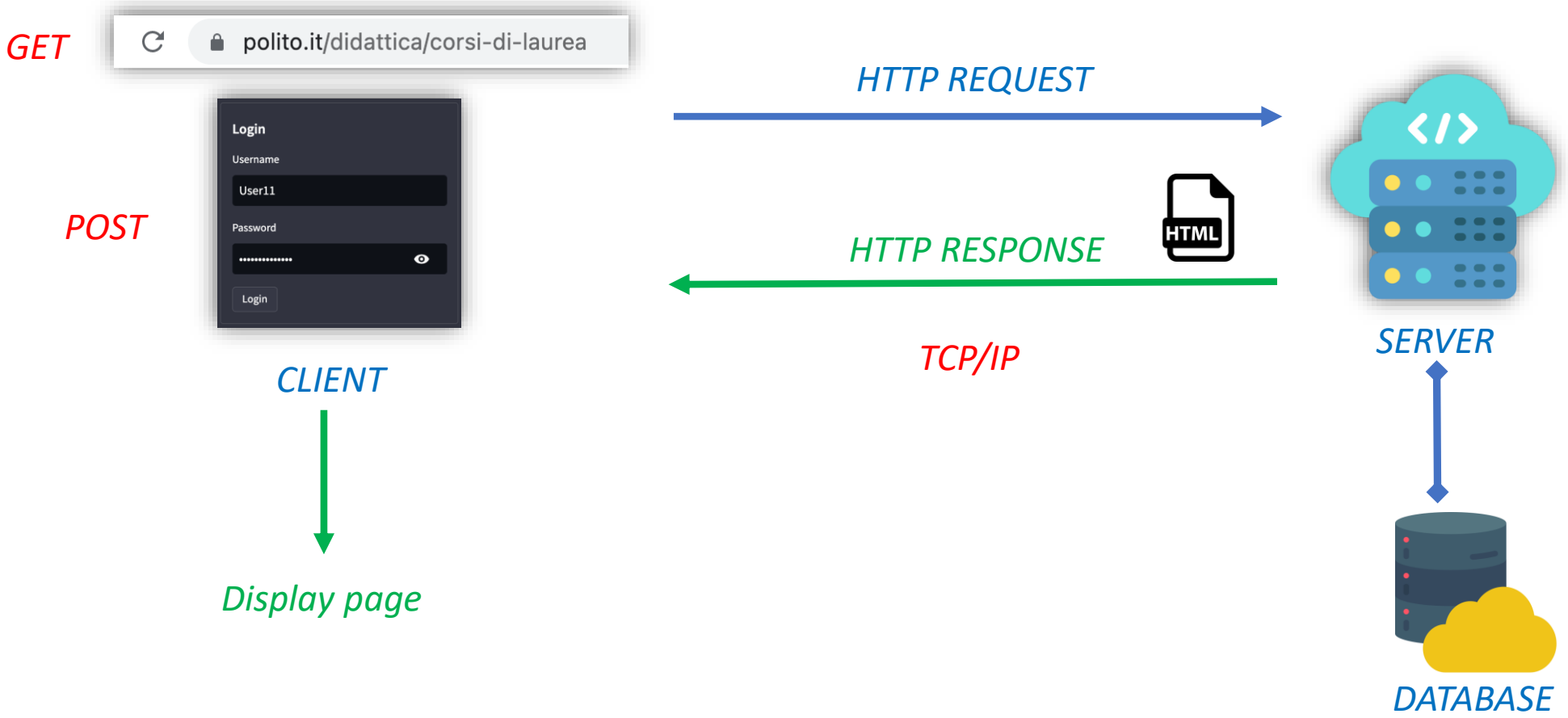
- **HyperText Transfer Protocol** (HTTP) It is the most widespread for communication between web-server and web-client, based on the *request-response* paradigm
- Used by the Client to make the request to the Server
- The request consists (but not exclusively) of a method and the URI, a path that uniquely identifies the resource

GET	POST	PUT	DELETE
Retrieve data	Insert data	Update data or insert it if it does not exist	Delete the specific resource

Main HTTP methods

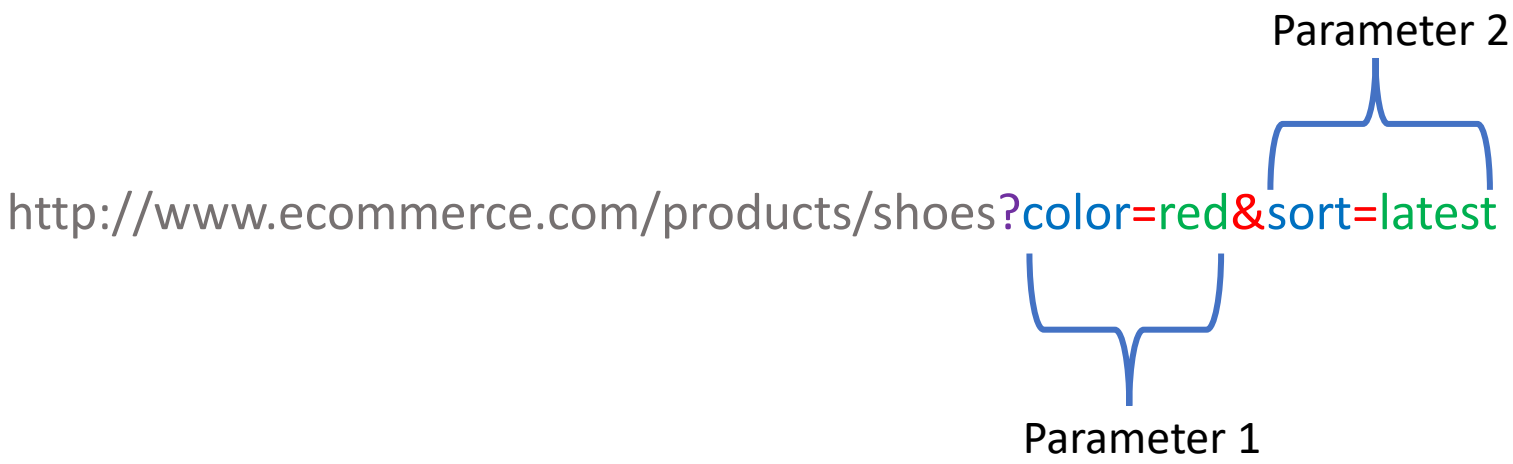
HTTP Protocol

- When the server receives the request, it processes it and eventually returns the desired resource (e.g., HTML page)



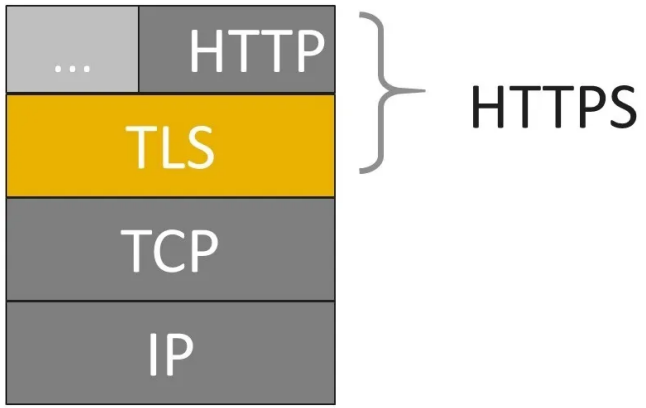
HTTP Protocol – URL Parameters

- URL parameters (also known as *query strings*) allow you to pass additional fields as web page input, to filter, complete queries, etc.
- Parameters start after the '?' character and are separated by '&'
- They are structured in the form *key = value*



HTTP Protocol – HTTPS

- HyperText Transfer Protocol Secure: protocol for secure, encrypted communication between client and server
- Use TLS/SSL to encrypt information based on a certificate that authenticates the web server
- Protects sensitive data from hacker attacks (*man in the middle, eavesdropping…)*
- New standard, where browsers report sites that don't use HTTPS as insecure



HTTP Protocol – Status Code

- In addition to executing the task and providing the requested resource, the Server returns a Status Code indicating the outcome of the request made.
- There are different types of messages, in turn grouped into 5 distinct classes:

1xx	2xx	3xx	4xx	5xx
Informative	Success	Redirection	Client-side error	Server-side error

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

Development

Web Applications

Web application development

- The development of a web application is divided into several components (“*separation of concerns*”), each with its own specific languages and tools

FRONTEND	**BACKEND**	**DATABASE**
The visible part of the application with which the user interacts directly.	*It takes care of server operation, request processing, and database access.*	*It deals with the storage of data, preferences and information.*
		

Web application development

- For both frontends and backends, there are different development *frameworks*, software libraries that provide a common basis for developing an application in a more efficient and organized way.

FRONTEND

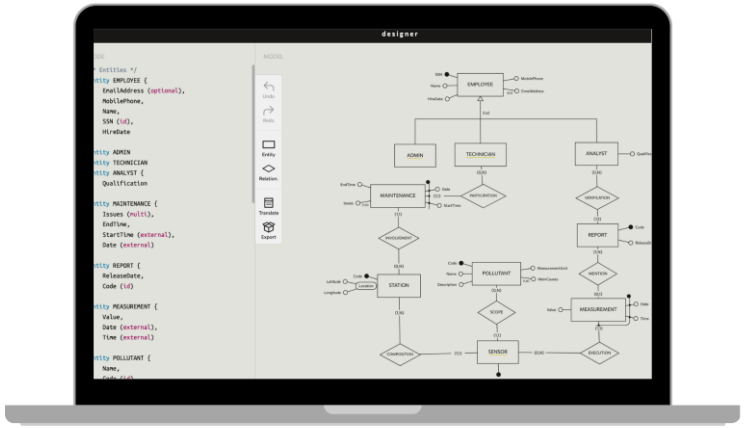
REACT

ANGULAR

VUE JS

FLUTTER

Example: designer was developed in Vue JS



designER

BACKEND

Express JS

python django

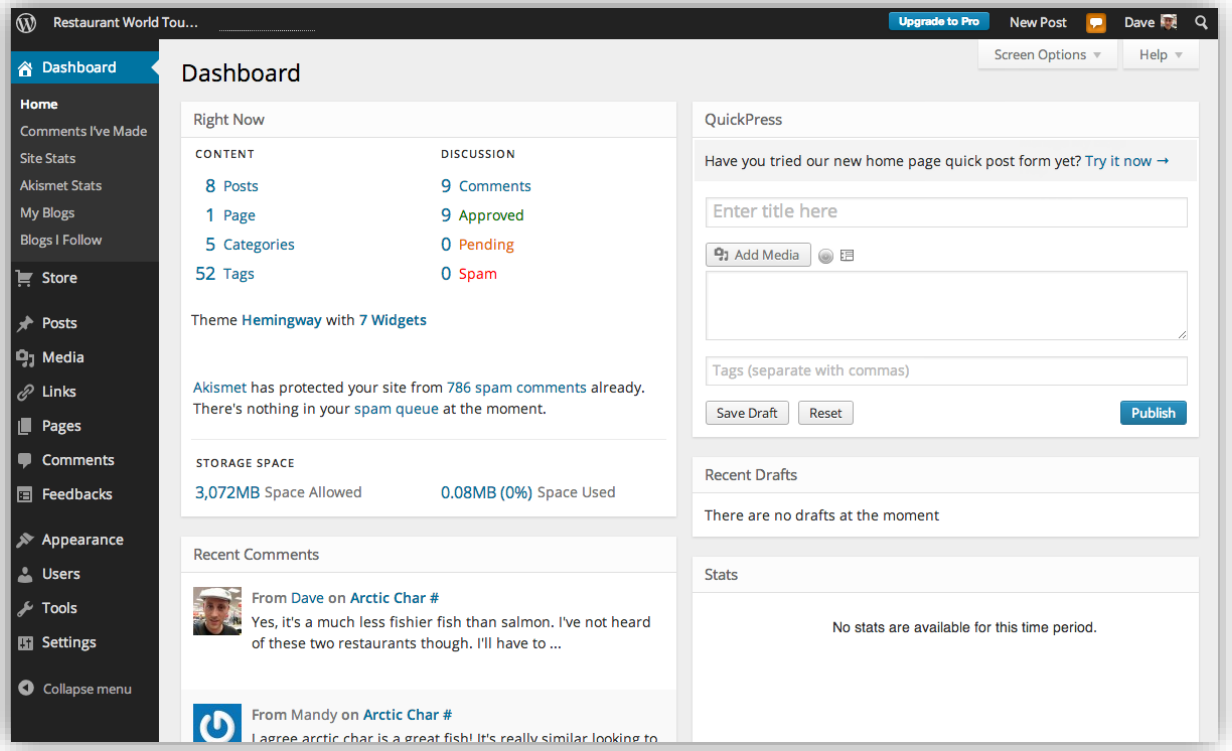
RAILS

php

Content Management System (CMS)

- Program for the management and creation of websites
- No need to write code or have special computer skills
- You can create websites, blogs, eCommerce, forums, etc...

- Popular: Wordpress, Wix, Drupal



The HTML language

- HTML: HyperText Markup Language
- Standard “de facto”
- Purpose: to provide a structured description of a program-independent hypertext document
- Used to create web pages, defining their structure, content and layout

Markup language <> Programming language

The HTML language - Tags

- HTML allows you to annotate a text to mark the parts that compose it through *tags*
- Tags are expressions that are always enclosed between the minor (<) and major (>) symbols
- Usually portions of text are delimited by pairs of tags (e.g., <h1>Title</h1>)
-

TAG	UTILIZZO
<head>	HTML document information
<body>	Contents of the web page
<h1>, **<h2>**, **<h3>**, **<h4>** […]	Section headings of different levels
<p>	Paragraph
<table>	Table

The HTML Language - Attributes

- Attributes can better characterize a tag
- attributes consist of a variable that is assigned a particular value

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF8-8">
    <title>Introduction to Databases</title>
</head>
<body style="background-color:powderblue;">
    <h1 style="color:red;">Introduction to Databases: Web Development</h1>
    <p style="font-family:courier">This is a sample HTML page. </p>

    <h2>What is HTML?</h2>
    <p>HTML is the acronym of HyperText Markup Language.</p>

    <h2>How to learn more about HTML?<h2>
    <p>There are many online resources available:</p>
    <ul>
        <li><a href="https://developer.mozilla.org/en-US/docs/Web/HTML">HTML: HyperText Markup Language | Mozilla Developer Network</a></li>
    </ul>

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

The HTML language - Overview

- HTML primarily allows you to create static web pages
- It allows you to include: images, audio, video, tables, forms, hyperlinks...
- Colors, fonts, backgrounds can also be managed by other languages such as ***CSS***
- To create dynamic web pages (created “on the fly” in response to user input) you use languages such as JS
- In some cases, the web page is dynamically generated server-side which returns the static page to the client.

Style Sheets

Introduced with HTML 4 for:

- enhance the description of presentation/style aspects
- allow separation between presentation and content

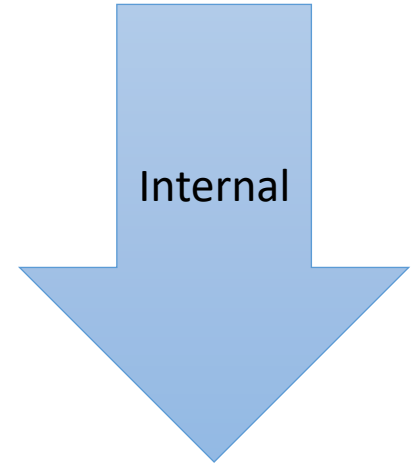
Style specifications can be indicated either in the HTML document or in separate files

- **In-line Style Specifications**: Style is specified directly within the HTML element
- **Internal Style Specifications**: the style is specified directly within the HTML document
- **External Style Specifications**: Style is specified outside the HTML document in separate style sheets

Style Sheets - Hierarchies

The CSS rule hierarchy indicates that the rules are applied in the following order:

1. Through an external style file (**external style**)
2. Tagged <style> internal to the document HTML (**internal style**)
3. Specifying style elements in any document tag using the style attribute (**in-line style**)



In case of conflicts, the most internal one is applied!

Style Sheets - Bootstrap

- Open-source front-end framework that provides a collection of ready-to-use CSS classes and JS functions

  Grid layouts, tables, forms, typography, panels, and more

- Follows the properties of responsive design

  Use CSS and HTML to resize, hide, contract, enlarge, or move web page content

- Makes web development fast and customizable

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

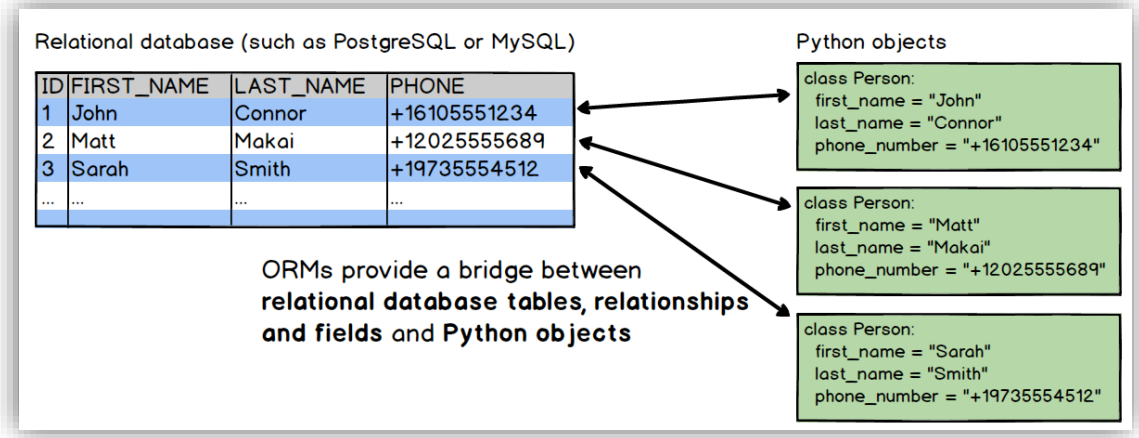
Python Backend Framework

For web development in Python there are different frameworks and micro-frameworks

Flask	CherryPy	python django
• Micro-framework • Jinja2 template • ORM handled by other packages • No admin interface • No authentication system	• Built-in tool for encoding, sessions, caching, authentication, static content • ORM handled by other packages	• Complex applications • Object-Relational Mapping (ORM) • Model-Template-View • Admin interface included • Built-in authentication system

Object-Relational Mapping (ORM)

- Integrate RDBMS systems within the object-oriented programming (**OOP**) paradigm
- Enables high-level abstraction to simplify development instead of using SQL directly
- **Portability** between DBMS
- Risk of performance degradation



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

Flask

- *Micro-framework* developed in Python to create web applications
- Includes a locally accessible web server
  - *localhost (http://127.0.0.1, http://localhost)*
  - *default port 5000*
- *Pythonic* approach, flexible and easy to learn

```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 is a high-level Python web framework that fosters rapid development and clean, pragmatic design[1]
- It is considered with “batteries included” as it is a complete framework of all the necessary elements for developers
- MVT design pattern (Model View Template)

Models	Views	Templates	URLs
Define data structure with database query mechanisms	Functions that receive an HTTP request and return the HTTP response	Describe the layout with which the results should be represented (e.g., HTML file)	Mapping to direct the HTTP request to the correct view
models.py	views.py	/templates	urls.py

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

Interaction with the DBMS

Web Applications

Overview

- A web application needs to interface with a database to perform queries
- Python has modules to interface with the main DBs: MySQL, PostgreSQL, Oracle, MariaDB...
- There is a common structuring of the interaction with the DBMS:
  1. Opening the connection with the DBMS
  2. Executing SQL queries
  3. Closing the connection

SQLAlchemy

- SQLAlchemy is a Python library that allows you to interface with a database efficiently
- Offers the flexibility and effectiveness of SQL within your Python application
- Supported features:
  1. Connecting to the DB
  2. Instant execution of SQL queries
  3. Data acquisition and reading
  4. Multiple queries and transactions

SQLAlchemy – Opening connection

- Starting point of applications that use SQLAlchemy, allows you to specify connection details
- Requires five parameters:
  1. **dialect**: name of the language that will be used for the connection
  2. **username**: username in DB
  3. **password**: user password
  4. **host**: name of the hosting machine the DBMS
  5. **dbname**: DB name

```
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
```

- Call to the function ***create_engine()***
- Connection to the DB with the ***connect()*** function

SQLAlchemy – SQL query

- Immediate execution of the statement: The server immediately compiles and executes the received SQL statement
- Call to the function ***execute()***
- Requests as a parameter the query to be executed, in string format
- If successful, it returns the result of the query, if it fails it raises an exception
- The result is stored with a variable of type "***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 – Transactions

- Connections occur implicitly in ***auto-commit*** mode

After the successful execution of each SQL statement, the commit is automatically performed.

- You must set up a ***non-automatic commit*** to execute it after a sequence of SQL statements

Only one commit is executed at the end of the execution of all statements.

SQLAlchemy – Transactions

- Call to the function ***begin()***
- When invoked, SQLAlchemy initializes a transaction and disables autocommit
- If successful, it returns an active transaction, otherwise it raises an **exception**

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

SQLAlchemy – Transactions

- If you disable autocommit, ***commit*** and ***rollback*** operations must be explicitly requested

commit()

- Commits the current transaction
- In case of failure raises an exception

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

rollback()

- Rolls back the current transaction
- In case of failure raises an exception

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

SQLAlchemy – Transactions

- If you disable autocommit, ***commit*** and ***rollback*** operations must be explicitly requested

- Using the construct ***with*** SQLAlchemy automatically handles *commit* or *rollback*
- *Commits if successful*
  - *If it fails, rolls back raises an exception*

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

SQLAlchemy – Closing connection

- Must be run when you no longer need to interact with the DBMS
- Closes the link with the DBMS and releases its resources
- Call to the function ***close()***

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