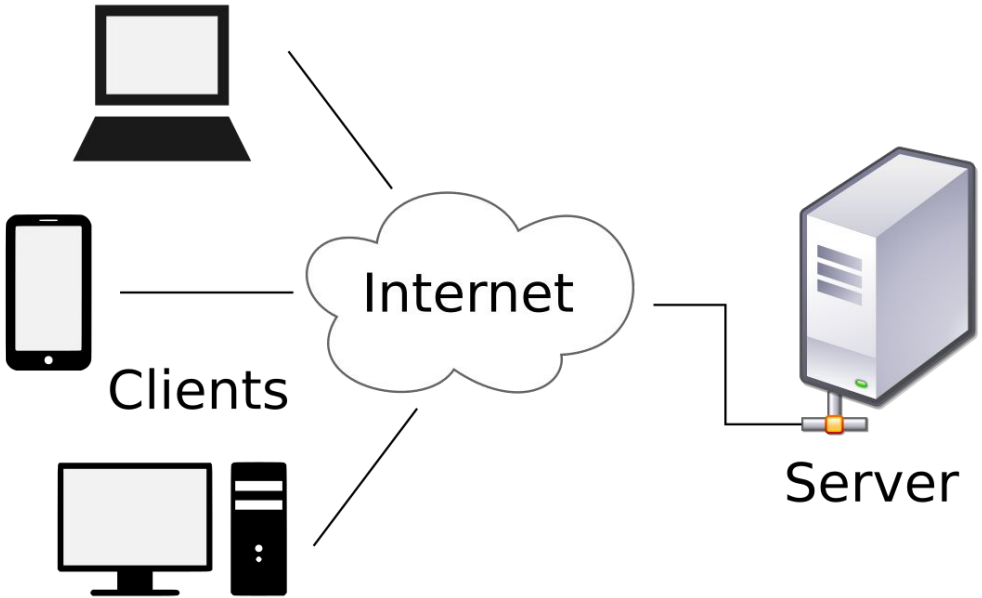
Processamento de Linguagem Natural

Luís Filipe da Costa Cunha
lfc@di.uminho.pt

José João Almeida
jj@di.uminho.pt

Word Wide Web

- Clients
- Servers
- Communication Protocol HTTP - Enables the exchange of information over the internet
- Web Pages
- Web Applications



Tim Berners-Lee, a British scientist, invented the World Wide Web (WWW) in 1989, while working at CERN …
It was developed to meet the demand for automated information-sharing between scientists in universities and institutes around the world.

Hypertext Transfer Protocol (HTTP) Requests

“An HTTP request is made by a client, to a named host, which is located on a server. The aim of the request is to access a resource on the server.”

- GET - request data from a specified resource
- POST - send data to the server
- PUT - send data to the server to update a resource
- DELETE - deletes the specified resource

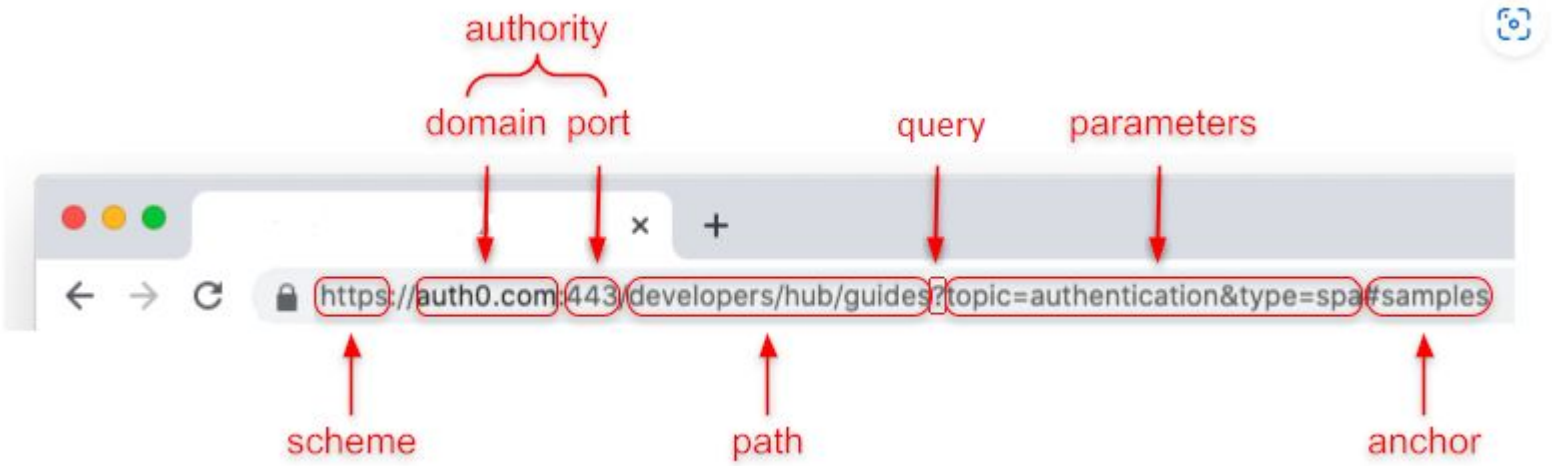
…

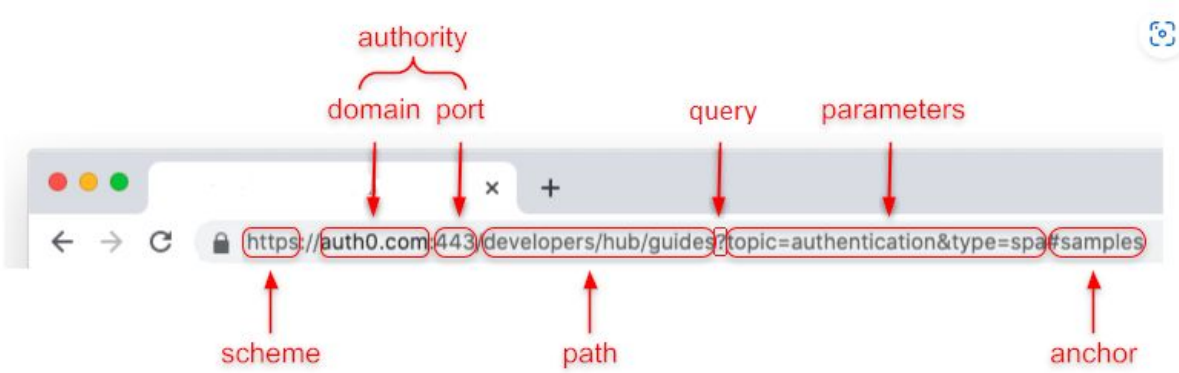
HTTP Codes

- Informational responses (100–199) → Request received, server is processing; Switching protocols (e.g., HTTP to WebSockets)
- Successful responses (200–299) → Resource successfully created (e.g., after a POST request).
- Redirection messages (300–399) → URL has changed permanently.
- Client error responses (400–499) → 404 Not Found
- Server error responses (500–599) → 500 Internal Server Error

Anatomy of a Uniform Resource Locator (URL)

A string that denotes the location of a given resource on the Internet: a web page, an image, a mailbox, etc.

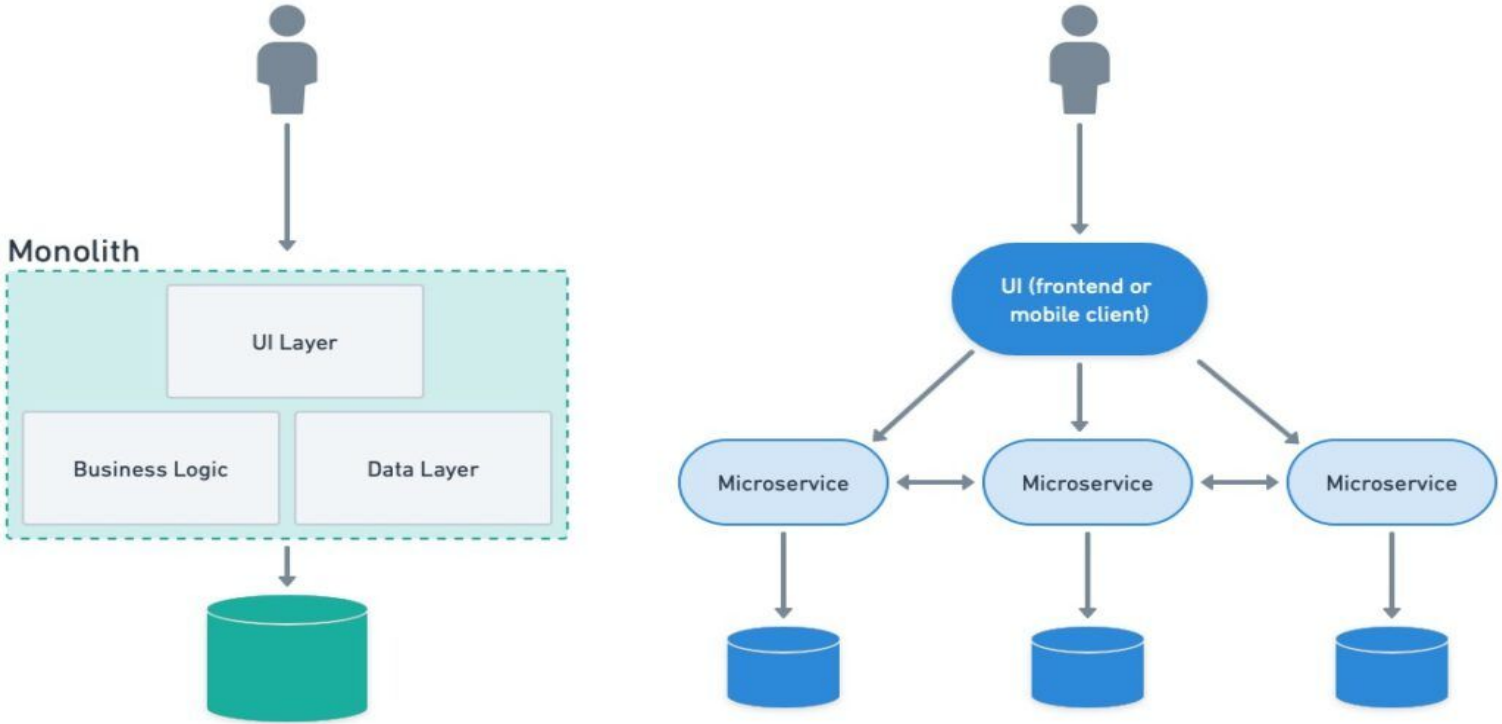




- **Scheme**: in a URL, this is the protocol that should be used to access the resource. Beyond the well-known HTTP and HTTPS, you can use many other schemes.
- **Domain**: this part indicates the server hosting the resource. It can be a domain name or an IP address.
- **Port**: it is the protocol port to which to send the request to access the resource. Usually, it is omitted, meaning that the default protocol port should be used.
- **Path**: this is the path to the resource on the hosting server.
- **Parameters**: these are optional extra information provided to the hosting server.
- **Anchor**: this part represents a specific part inside the resource. It is also called fragment.

Examples

- http://jwt.io
- https://auth0.com/docs/get-started#learn-the-basics
- https://identicons.dev/static/icons/mono/png/icon-access-token.png
- mailto:yourfriend@somewhere.com
- ftp://ftpserver.com/myfolder


Monolith
UI Layer
Business Logic
Data Layer
UI (frontend or mobile client)
Microservice
Microservice
Microservice

- **Amazon** — over 1000 microservices
- **Netflix** — over 700 microservices
- **Uber** — over 500 microservices


Uber
As a result, we adopted a microservice architecture. Ultimately our systems became more flexible, which allowed teams to be more autonomous.
Uber
Uber Technologies Inc.

FLASK

- Develop web applications easily
- Jinja2 template engine
- Small learning curve
- pip install flask

FLASK

```python
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"

app.run(host="localhost", port=4002, debug=True)
```

```python
@app.route('/api/conceitos)
def conceitos_api():
    db = json.load(dbFile)
    return db
```

“Routing or router in web development is a mechanism where HTTP requests are routed to the code that handles them”

FLASK

```
@app.route('/conceitos')
def conceitos():
    termos = list(db.keys())
    return termos

@app.route('/conceitos/<designacao>')
def conceito(designacao):
    desc = db[designacao] #key error
    return {'designacao': designacao, 'descricao': desc})
```

FLASK

```python
from flask import Flask, request

#@app.route("/conceitos", methods = ["POST"])
@app.post("/conceitos")
def adicionar_conceito():
    #json
    data = request.get_json()
    #form data
    designacao = request.form.get("designacao")
    descricao = request.form.get("descricao")
    ...
    return

#@app.route("/conceitos/<designacao>", methods = ["DELETE"])
@app.delete("/conceitos/<designacao>")
def delete_conceitos(designacao):
    ...
    return
```

FLASK

```
from flask import Flask, render_template, request

@app.route('/conceitos')
def conceitos():
    termos = list(db.keys())
    return render_template('conceitos_view.html', title=Conceitos, termos = termos)

@app.route('/conceitos/<designacao>')
def conceito(designacao):
    desc = db[designacao] #key error
    return render_template('conceito_view.html', dados = {'designacao': designacao, 'desc': desc})
```

Jinja2 - Template Engine

“Jinja is a fast, expressive, extensible templating engine. Special placeholders in the template allow writing code similar to Python syntax. Then the template is passed data to render the final document.”

Jinja2 - Parent template


```
<!DOCTYPE html>
<html>
<head>
    {% block head %}{% endblock %}
    <meta charset="utf-8">

</head>
<body>

    {% include 'header.html' %}
    <div class="container mt-5">
        {% block body %}
        {% endblock %}
    </div>

    {% include 'footer.html' %}

</body>
</html>
```

child templates can fill the parent blocks

return the rendered content of the included file

Jinja2 - child template


```
{% extends 'layout.html' %}
```

first locates the parent template

```
{% block head%}
<title>{{title}}</title> (important for search engine optimization (SEO))
{% endblock %}
```

```
{% block body%}
```

fill the parent blocks

```
<h1>Conceitos</h1>
<ul>
    {% for conceito in conceitos %}
        <li><a href="/termos/{{conceito}}">{{conceito}}</a></li>
    {% endfor %}
</ul>
{% endblock %}

### {% if conceitos|length == 0 %} …. {% elif conceitos|length < 10 %} … {% else %} … {% endif %}

https://jinja.palletsprojects.com/en/3.1.x/templates/
```

Processamento de Linguagem Natural

Luís Filipe da Costa Cunha
lfc@di.uminho.pt

José João Almeida
jj@di.uminho.pt