Despliegues de sistemas de ML orientado a microservicios

Clase 3

FAMAF Facultad de Matemática, Astronomía, Física y Computación
UNC
Córdoba Technology Cluster
CCAD Centro de Computación de Alto Desempeño de la Universidad Nacional de Córdoba

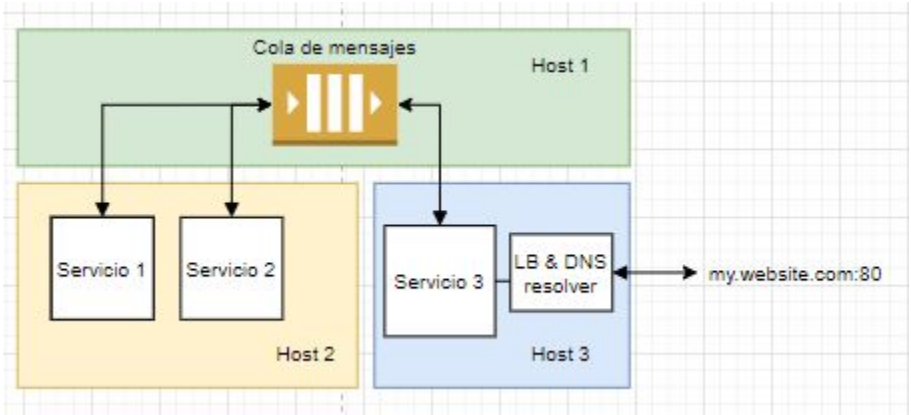
Temas clase 3

- Frameworks web para desarrollo de APIs
  - Stacks
  - Datos y ORM
  - Autenticacion y ~~autorizacion~~
- Ingeniería de software aplicada a APIs de modelos de ML
  - Diseño
  - Retroalimentación para mejorar precisión

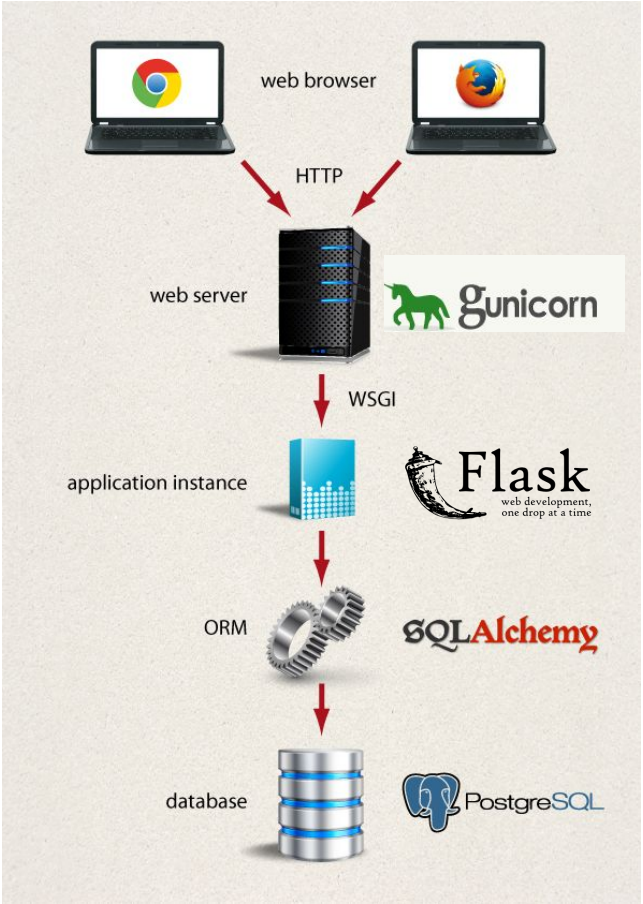
- Actividad: ML API

Frameworks para aplicaciones web

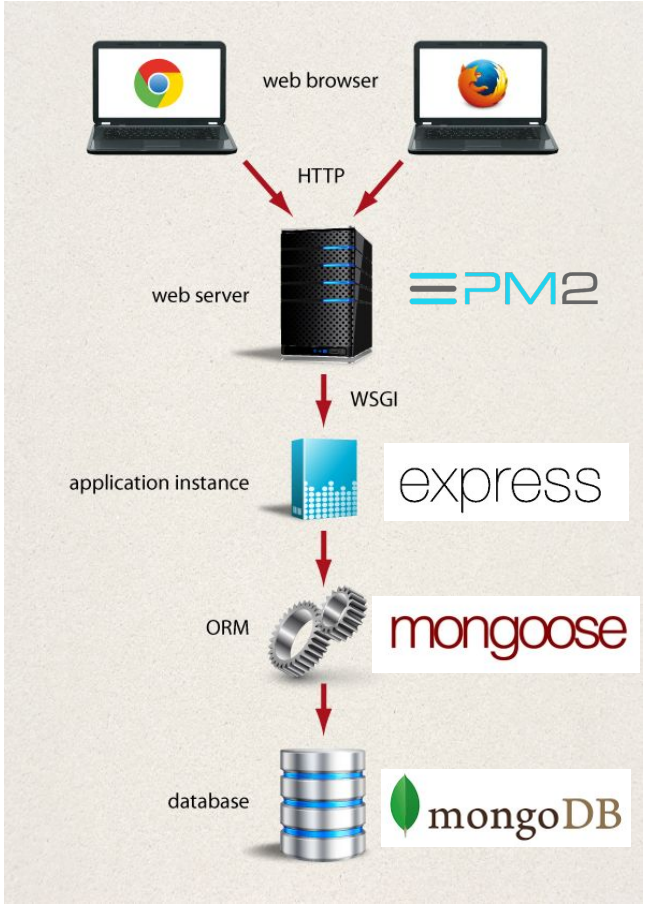
Hasta el momento tenemos entonces



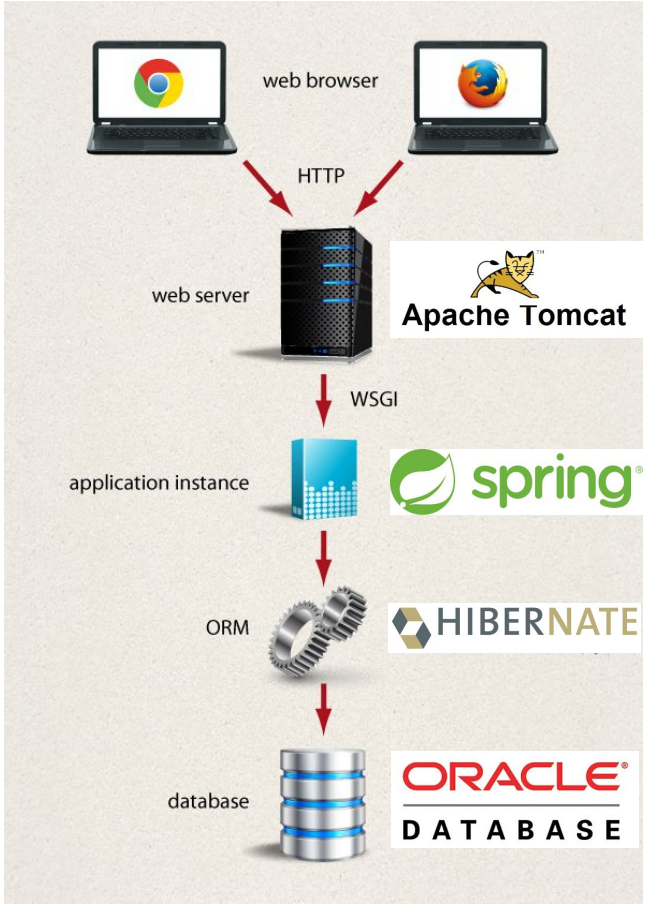
Estructura de una app web



Estructura de una app web



Estructura de una app web



Introducción a flask servidor de desarrollo

hello.py

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

app = Flask(__name__)

@app.route('/')
def hello():
    name = request.args.get("name", "World")
    return f'Hello, {escape(name)}!'
```

```
$ env FLASK_APP=hello.py flask run
 * Serving Flask app "hello"
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Introducción a flask servidor de producción

hello.py

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

app = Flask(__name__)

@app.route('/')
def hello():
    name = request.args.get("name", "World")
    return f'Hello, {escape(name)}!'
```

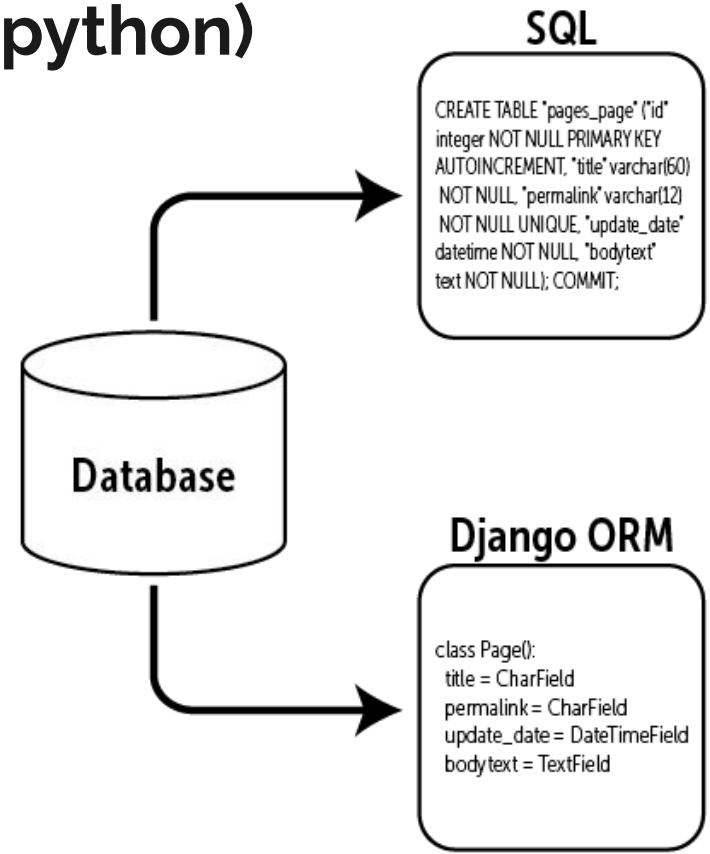
```
$ gunicorn -w 4 hello:app
```

Object Relational Mapping (python)

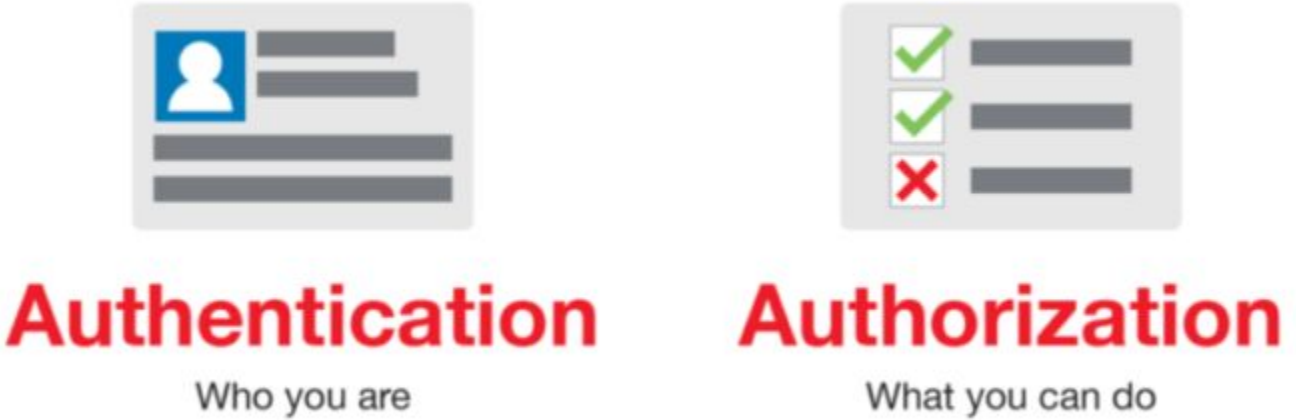
peewee

SQLAlchemy

PONY



Autenticación y autorización



Autenticación de APIS

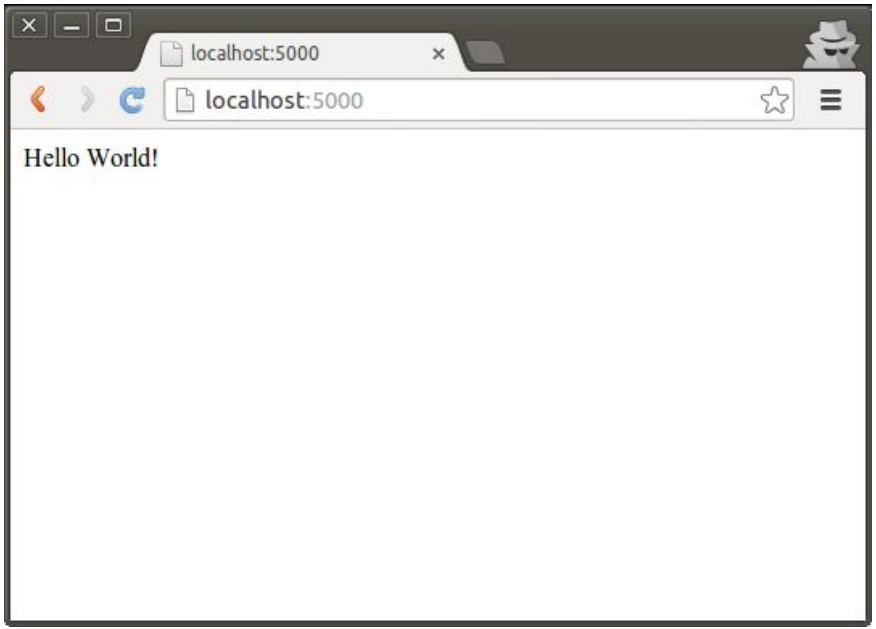
- BasicAuth: Envio de usuario y contraseña en cada request.

```python
from flask import Flask, render_template
from flask_basicauth import BasicAuth

app = Flask(__name__)

app.config['BASIC_AUTH_USERNAME'] = 'john'
app.config['BASIC_AUTH_PASSWORD'] = 'matrix'

basic_auth = BasicAuth(app)

@app.route('/secret')
@basic_auth.required
def secret_view():

    return render_template('secret.html')
```

- TokenAuth: Envía header **Authorization: Bearer <TOKEN>**
- SessionAuth: Envío de **cookie** de session ID.

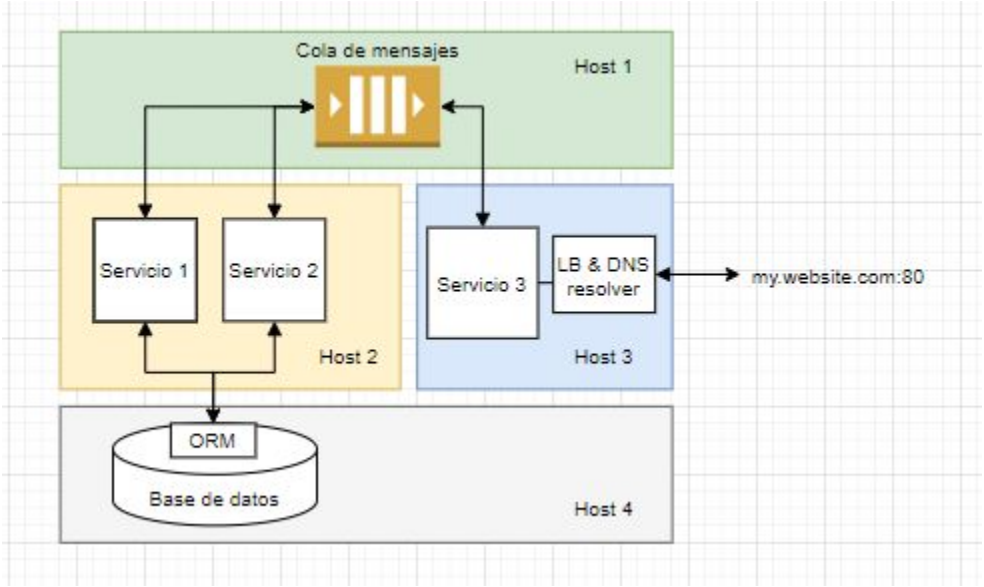
Proyecto de autenticacion en flask con login

Flask “Hello World”



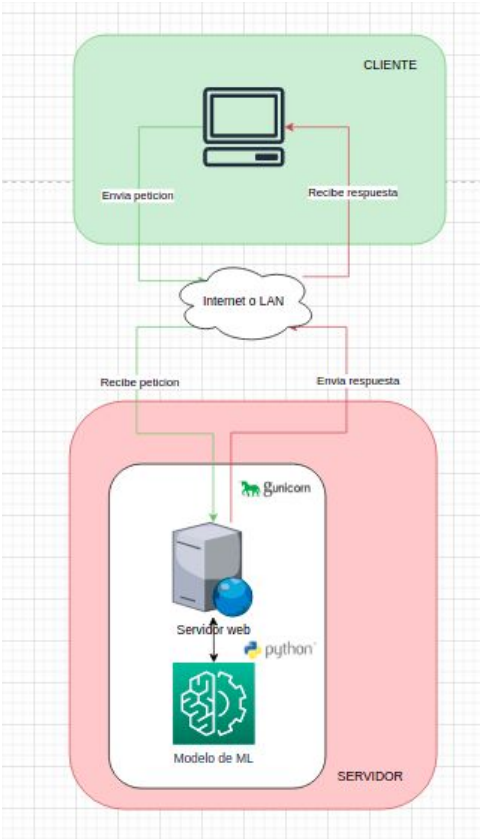
https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world

Nuestro sistema distribuido quedaría entonces



Ingeniería de software aplicada a API de ML

Arquitectura Simple



Arquitectura Simple

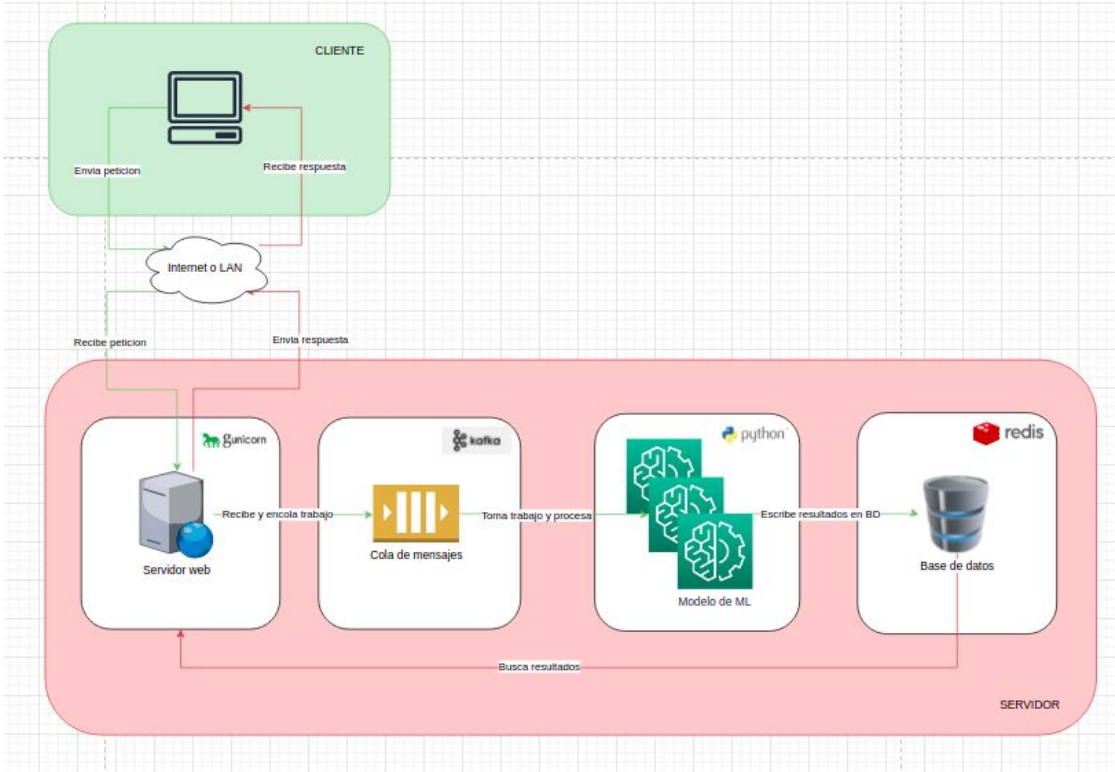
Pros

- Fácil de implementar
- Útil para desplegar rápidamente POCs

Cons

- API y modelo de ML juntos (no permite escalar alguna de las 2 partes)
- Que ocurre con datos más pesados (ej. Imágenes)?

Arquitectura mejorada



Arquitectura mejorada

Pros

- Permite escalar cada componente por separado de forma sencilla
- Se puede actualizar cada componente independientemente
- Permite agregar fácilmente herramientas adicionales a nuestro pipeline

Cons

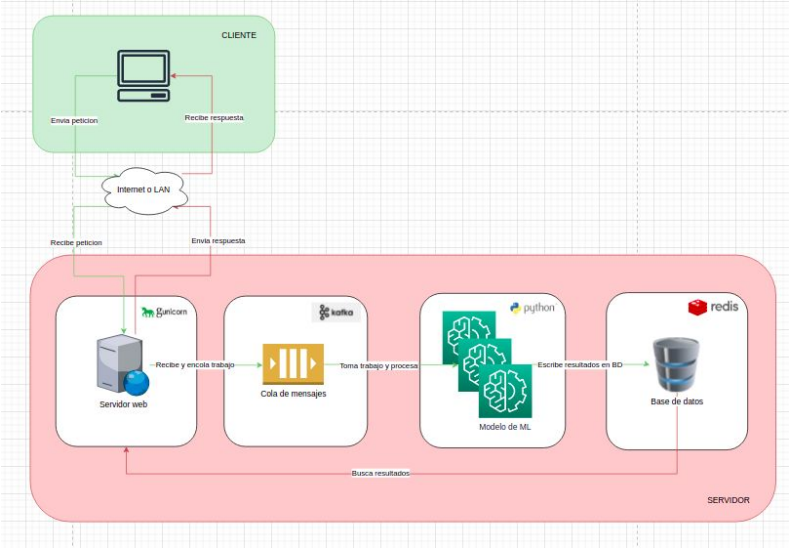
- Requiere el uso y conocimiento de herramientas adicionales
- Una mala configuración puede empeorar nuestra perfomance significativamente
- Más difícil de debuggear, necesitamos una suite de tests robustos

Actividad 2

API de ML

Actividad 2

- Vamos a desplegar una API que brindara servicios de análisis de sentimiento en texto
- Usaremos: *Docker*, *Docker-Compose*, *Flask*, *Gunicorn*, *Redis*, *Kafka y Scikit-Learn*
- La arquitectura de nuestra API será la misma presentada previamente



Fork from: https://github.com/matisilva/ml-api-public

<FIN DE CLASE>