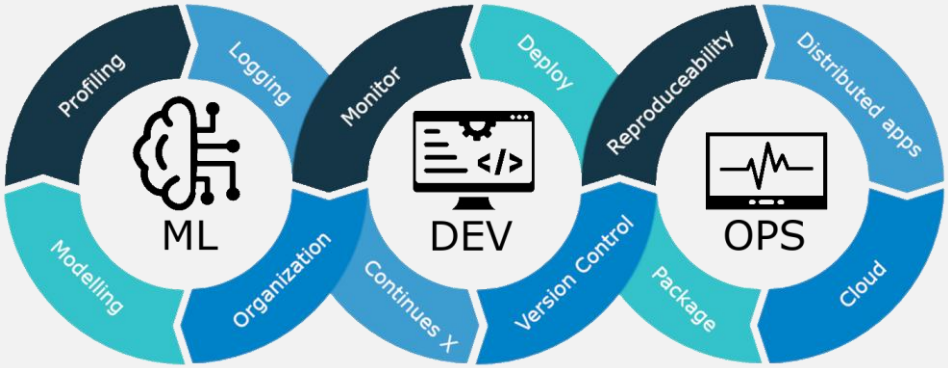
FastAPI

Créer des API avec des performances élevées et une documentation automatique.

UP ASI
Département informatique
Bureau: E204

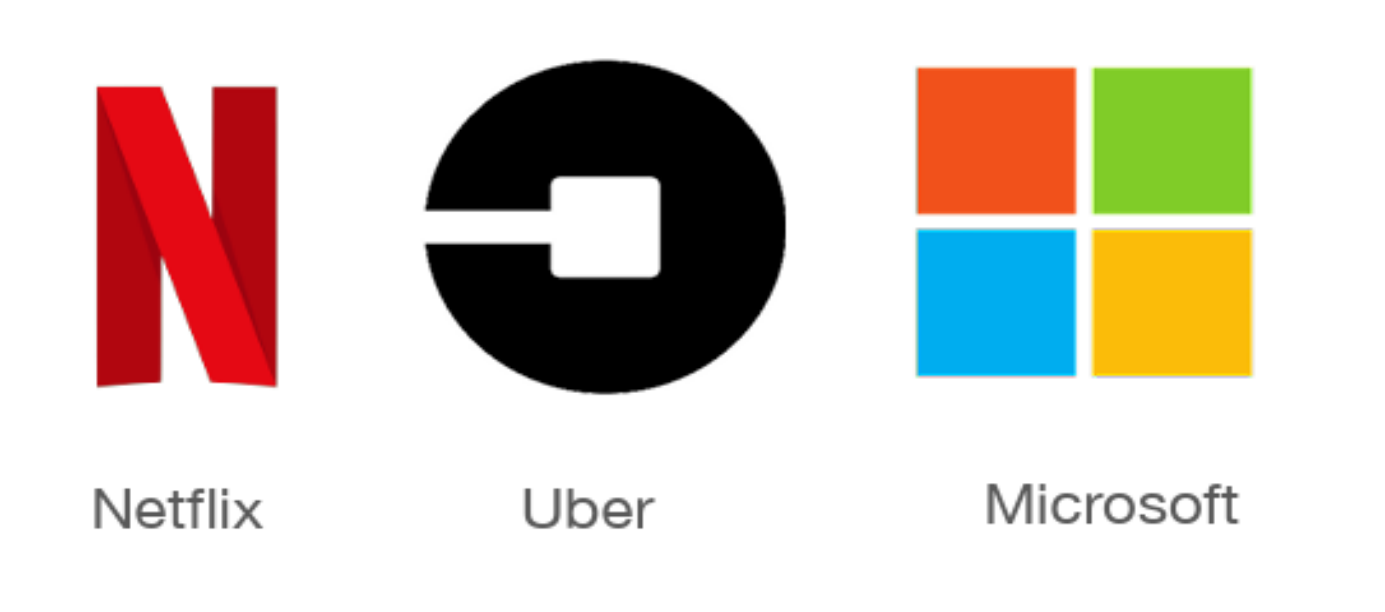


Définition

- FastAPI est un framework web pour Python, conçu pour créer des APIs modernes et rapides.
- FastAPI est conçu pour maximiser la productivité et minimiser les erreurs.
- FastAPI est un choix puissant pour déployer des modèles ML en production.
- Grâce à sa simplicité et ses performances, il s'intègre parfaitement dans un workflow MLOps.

Définition

- FastAPI est largement utilisé chez les entreprises



FastAPI dans MLOps

Contexte de MLOps :

- ✓ Besoin de déployer rapidement des modèles ML en production.
- ✓ Importance de l'exposition des modèles via des APIs.

Avantages de FastAPI pour MLOps :

- ✓ Documentation interactive automatique pour tester les modèles.
- ✓ Haute performance pour servir des prédictions rapidement.
- ✓ Intégration simple avec d'autres outils (Exemple : MLflow)
- ✓ Pour consommer les web services fournis par FastAPI, vous pouvez utiliser plusieurs technologies frontales (Angular , React, Flutter, etc…) pour avoir une interface conviviale et simple pour le client final.

FastAPI dans MLOps

Caractéristiques principales:

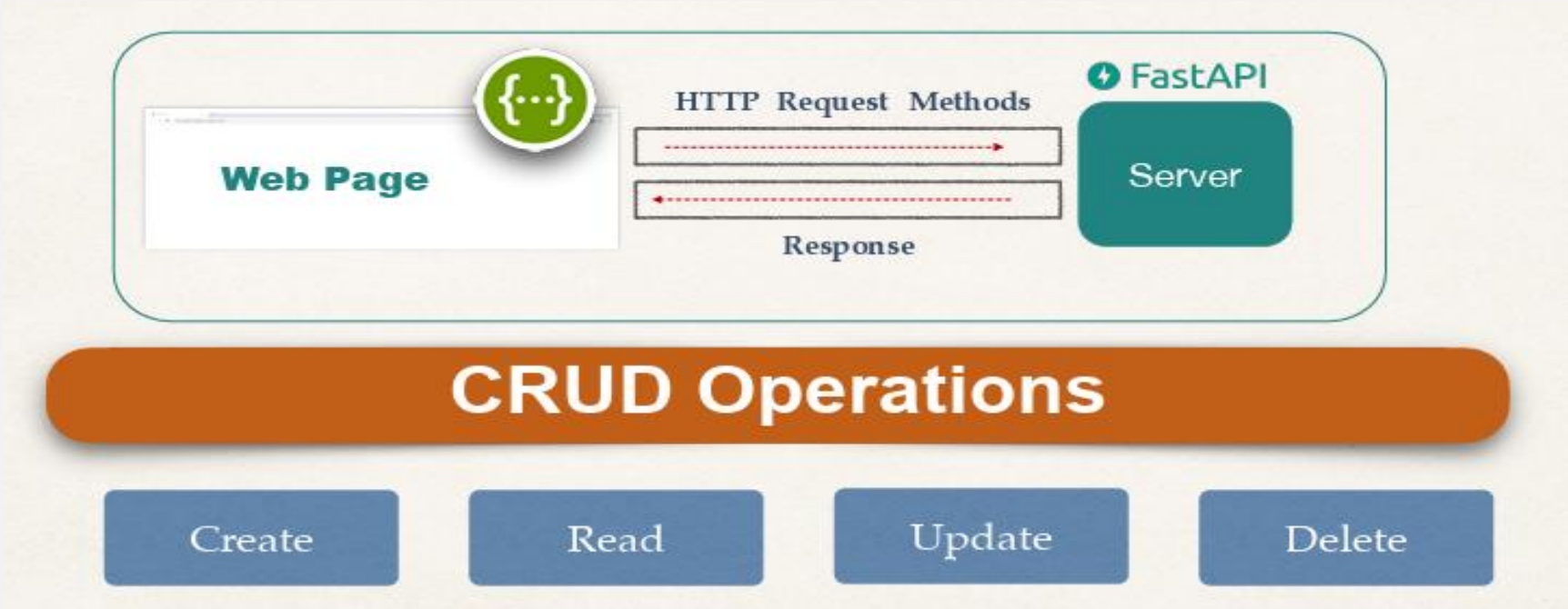
- ✓ **Rapide et performant :** FastAPI permet de créer des applications qui répondent rapidement, ce qui est idéal pour des services modernes.
- ✓ **Documentation automatique :** FastAPI génère une page web interactive où vous pouvez tester vos APIs sans écrire de code supplémentaire.
- ✓ **Compatible avec les standards :** Il utilise des règles universelles comme OpenAPI pour décrire les APIs et JSON Schema pour vérifier les données.

Pourquoi FastAPI

Caractéristique	FastAPI	Flask	DJANGO	TORNADO
Performance	Très rapide	Relativement lent	Modéré	Très rapide (écrit en C ,applications à haute performance)
Documentation automatique	Génération automatique via Swagger UI et ReDoc	Pas de documentation automatique intégrée	Documentation automatique via Django REST framework	Pas de documentation automatique intégrée
Communauté	Très grande	Très grande	Très grande	Plus petite
Prise en charge des API REST	Parfaitement adapté	Très bien adapté	Excellente prise en charge (Django REST framework)	Support des APIs REST (application temps réel)

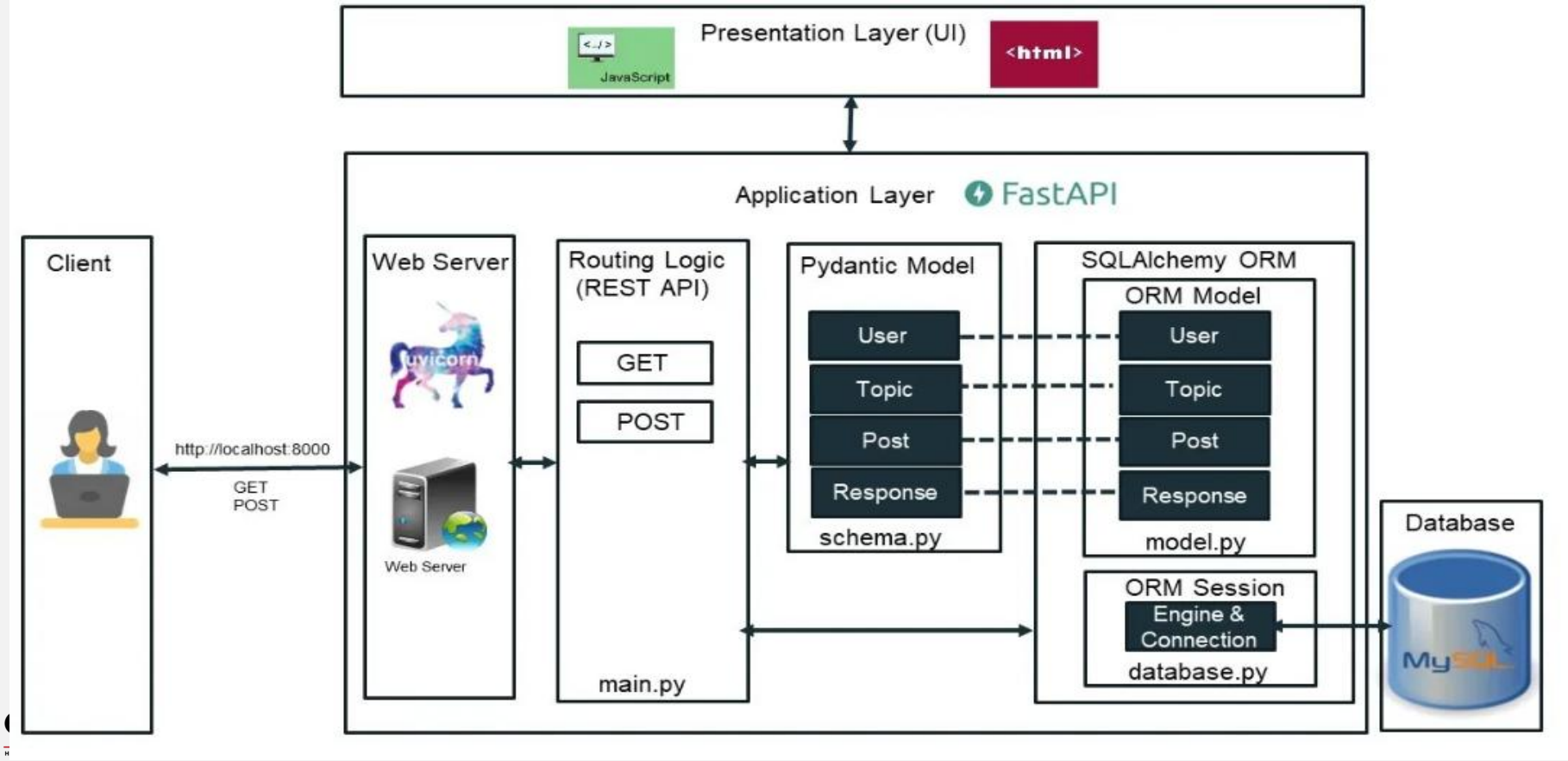
Architecture FastAPI

FastAPI est utilisé pour exécuter des opérations tels que les CRUD qui s'exécute sur son serveur uvicorn et dont les résultats sont affichés avec une documentation grâce à Swagger.



Architecture FastAPI

FastAPI s'appuie sur le serveur web Uvicorn qui est un serveur web ASGI (Asynchronous ServerGateway Interface) rapide pour Python qui est idéal pour le développement. L'architecture complète d'un projet python utilisant FastAPI est la suivante :



Exemples d'utilisation

Exemple1 : création d'un end point avec FastAPI et pydantic

```python
from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI()

class Item(BaseModel):
    code: str
    description: str = None
    totalPrice: float

@app.post("/items/")
def create_item(item: Item):
    return {"code": item.code, "totalPrice": item.totalPrice}
```

Exemples d'utilisation

Exemple2 : entrainer un modèle en utilisant la regression linéaire

```
import joblib
import numpy as np
from sklearn.linear_model import LinearRegression

# entrainer le modele sur la taille de la maison en
# metres carres et nombre de chambres
X = np.array([[60, 3], [80, 4], [100, 3], [120, 4], [150, 5], [200, 5]])
# Prix des maisons en fonction des donnees deja saisies
y = np.array([300000, 400000, 500000, 600000, 750000, 900000])

model = LinearRegression()
model.fit(X, y)

# Sauvegarder le modele
joblib.dump(model, 'model.joblib')
```

Exemples d'utilisation

Exemple2 : création d'un end point pour la prédiction du modèle entrainé

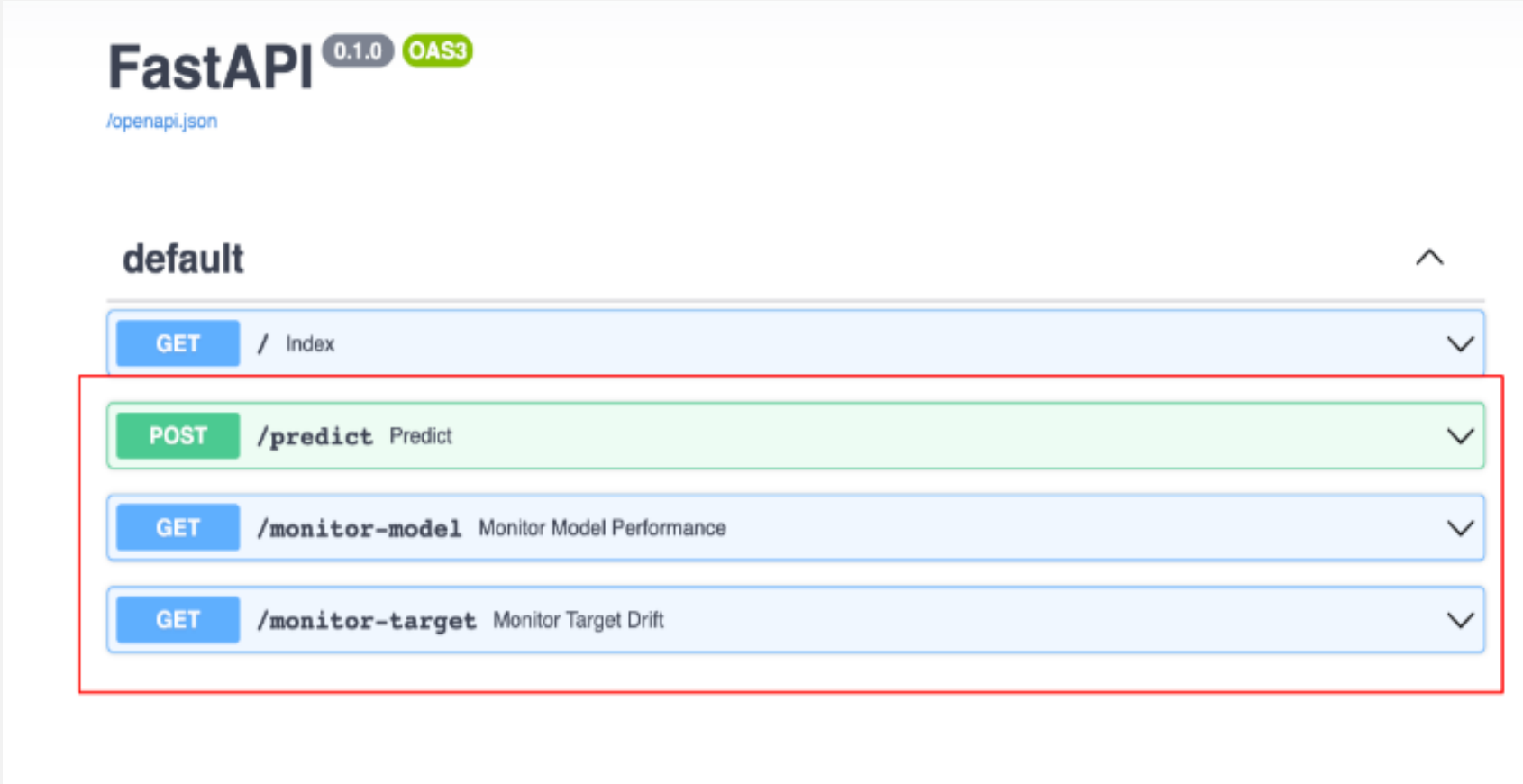
```
from fastapi import FastAPI
from pydantic import BaseModel
import joblib
import numpy as np

model = joblib.load('model.joblib')
app = FastAPI()

class HouseFeatures(BaseModel):
    size: float  # Taille de la maison en metre carre
    rooms: int   # Nombre de chambres

@app.post("/predict/")
def predict_price(features: HouseFeatures):
    X_new = np.array([[features.size, features.rooms]])
        prediction = model.predict(X_new)
    return {"predicted_price": prediction[0]}
```

FastAPI Ui


FastAPI
0.1.0
OAS3
/openapi.json
default
GET / Index
POST /predict Predict
GET /monitor-model Monitor Model Performance
GET /monitor-target Monitor Target Drift

FastAPI Ui


Curl
curl -X 'POST' \
'http://127.0.0.1:8000/predict' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
"sepal_length": 7,
"sepal_width": 3,
"petal_length": 8,
"petal_width": 8
}'
Request URL
http://127.0.0.1:8000/predict
Server response
Code
Details
200
Response body
{
"predicted_class": 2,
"predicted_class_name": "virginica"
}
Download
Response headers
content-length: 56
content-type: application/json
date: Thu,08 Aug 2024 10:26:02 GMT
server: uvicorn

Prochaines étapes

→ Pratiquez avec le workshop fourni et explorez les fonctionnalités avancées (comme l'intégration avec MLflow).

FastAPI

Si vous avez des questions, n'hésitez pas à nous contacter :

Département Informatique

UP Architectures des Systèmes d'Information

Bureau E204 /E304