



Programmation back-end

Fabien Coelho, Claire Medrala

Mines Paris – PSL

Avril 2024



- 1 Architecture
- 2 REST
- 3 DB API
- 4 AnoDB
- 5 Flask
- 6 Conseils
- 7 CRUDS
- 8 AAA
- 9 Blueprint
- 10 Déploiement



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Architecture back-end



Architecture back-end

très simplifiée

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

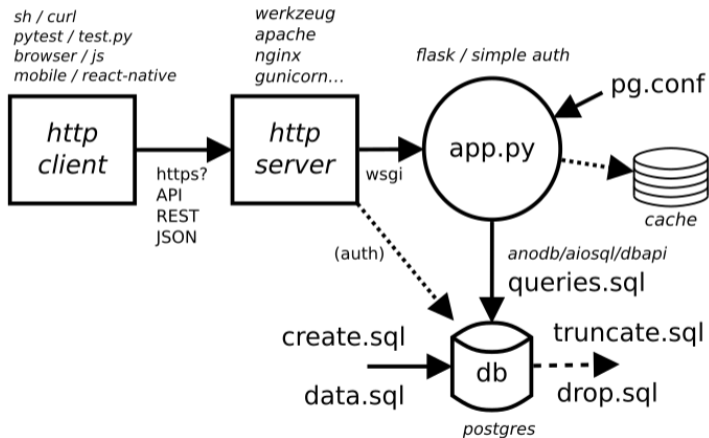
Conseils

CRUDS

AAA

Blueprint

Déploiement





Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

HTTP	<i>HyperText Transport Protocol</i>	RFC 2616
WSGI	<i>Python Web Server Gateway Interface</i>	PEP 333
ASGI	<i>Python Asynchronous Server Gateway Interface</i>	
Flask	requête HTTP : conf, params, auth, réponses, json. . . <i>app</i> <i>votre application !</i>	
AnoDB	surcouche de AioSQL qui gère les connexions	
AioSQL	gestion de requêtes à une base de donnée relationnelle	
DB API	interface vers une base de données relationnelle	PEP 249
Postgres	base de donnée relationnelle (ou tests avec SQLite)	
Redis	cache de données (clef-valeur), partagé	



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Préoccupation globale !

Authentification

Qui ?

- utilisateur, mot de passe. . .
- HTTP Basic, Digest, paramètres, *token*

Authorization

Quoi ?

- utilisateur, rôle (groupe). . .
- par route ou selon les données

Audit

Quand ?

- qui fait quoi : traces, données, historique



Architecture back-end . . .

Exemple AWS

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

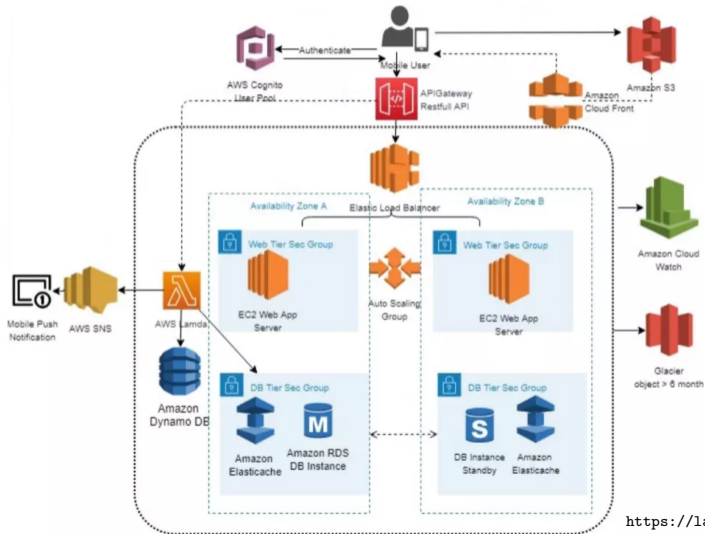
Conseils

CRUDS

AAA

Blueprint

Déploiement



<https://labs.sogeti.com/>



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

REST



API REST

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

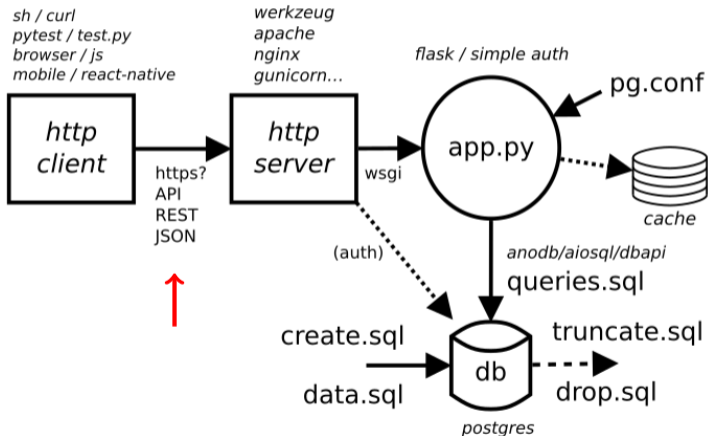
Conseils

CRUDS

AAA

Blueprint

Déploiement





REST – REpresentational State Transfer

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

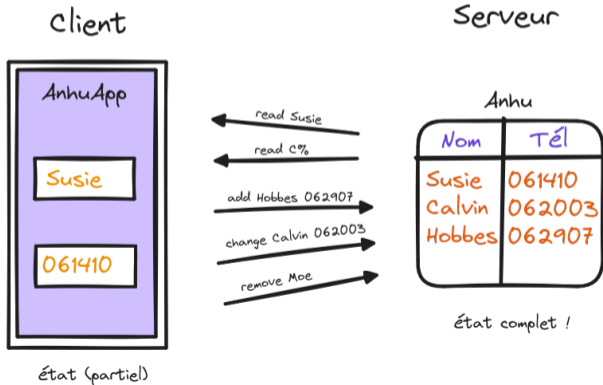
Déploiement

Architecture logicielle

Interface

■ transfert d'états

identification et manipulation de ressources





REST – REpresentational State Transfer

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Architecture

- proposé par Roy Fielding (HTTP, Fondation Apache)
PhD à Irvine (Ca) en 2000 sur Architecture du Web...
- s'appuie sur le protocole HTTP

Interfaces avec 5 contraintes principales

API

client-serveur asymétrique, requête-réponse

sans état requêtes auto-contenues

uniforme forme URI, paramètres et résultats en JSON (ou XML)

cachable informations gardées par le client

en couche scalability avec proxy, équilibrage de charge...



HTTP – protocole client-serveur

`https://calvin:mdp@kiva.mobapp.minesparis.psl.eu/api/store?filter=c%`

Requête HTTP

```
GET /api/store HTTP/1.1
Host: kiva.mobapp.minesparis.psl.eu
Authorization: Basic Y2FsZm1uOmkcA==
Content-Length: 16
Content-Type: application/json

{"filter": "c%"}
```

méthode GET POST...
chemin /api/store
entêtes eg authentication, ...
corps données, eg JSON

Réponse HTTP

```
HTTP/1.1 200 OK
Server: Apache/2.4.52 (Ubuntu)
Content-Length: 34
Content-Type: application/json

[{"key": "calvin", "val": "hobbes"}]
```

code 2xx 4xx 5xx...
état OK, Error...
entêtes eg serveur, date, ...
corps données, eg JSON

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement



REST / HTTP – concrètement

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Appel de fonction, avec effet de bord, à travers HTTP

chemin identification d'une ressource

méthode GET POST PATCH PUT DELETE

idempotence GET PUT DELETE, cachabilité GET POST

paramètres HTTP ou JSON, retour JSON

authn authentification HTTP basic/param, token...

Requête

```
GET /students/5432 HTTP/1.1
```

```
Host: api.minesparis.psl.eu
```

```
Authorization: Basic Zm9vOmJsYT8h
```

Réponse

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json
```

```
Content-Length: 31
```

```
{"nom": "Calvin", "classe": "CE1"}
```



REST / HTTP – conventions

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Méthodes

SQL

GET consultation

SELECT

POST ajout (retour *id* ?)

INSERT

PATCH/PUT modification (partielle/totale)

UPDATE

DELETE effacement

DELETE ?

Chemin – **identification** d'une ressource

tuple(s)

`/students` liste des étudiants

`/students/42` l'étudiant numéro 42

`/students/42/courses` liste des cours de l'étudiant 42

`/courses/53` le cours 53

`/courses/53/students` liste des étudiants du cours 53



REST / HTTP – paramètres

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Localisation

implicite	dans le chemin lui-même	/courses/42
explicite	en HTTP ou JSON ou XML	nom=...
valeur	texte, éventuellement convertie	...=2020-07-29

Usage

- identification de la ressource concernée chemin implicite
- valeurs (données, critères de recherche)... explicite

GET /students	nom=C%	<i>dont le nom commence par C</i>
POST /students	nom=Hobbes ne=...	<i>création de Hobbes</i>
PATCH /students/42	ne=2001-02-03	<i>modification date de naissance</i>
PUT /students/42	ne=2001-02-02 nom=...	<i>modification complète</i>
DELETE /students/42		<i>efface cet étudiant</i>
DELETE /students		<i>efface tous les étudiants</i>



REST / HTTP – exemples de requêtes et réponses

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Paramètres JSON

```
GET /api/store HTTP/1.1
Host: kiva.mobapp.minesparis.psl.eu
Authorization: Basic Y2FsdmluOmlkcA==
Content-Length: 16
Content-Type: application/json

{"filter": "c%"}
```

Paramètres HTTP

(bof)

```
GET /api/store HTTP/1.1
Host: kiva.mobapp.minesparis.psl.eu
Authorization: Basic QXJlIHlvdSBqb2tpbmc/
Content-Length: 11
Content-Type: application/x-www-form-urlencoded

filter=c%25
```

Réponse JSON liste de tuples

```
HTTP/1.1 200 OK
Server: Apache/2.4.52 (Ubuntu)
Content-Length: 22
Content-Type: application/json

[["calvin", "hobbes"]]
```

Réponse JSON liste de dicts

```
HTTP/1.1 200 OK
Server: Apache/2.4.52 (Ubuntu)
Content-Length: 34
Content-Type: application/json

[{"key": "calvin", "val": "hobbes"}]
```




Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

HTTP Status Codes

200	OK	GET
201	Created	POST
204	No Content	DELETE PATCH PUT
400	Bad Request	eg pb paramètres ?
401	Unauthorized	authentification
403	Forbidden	autorisation
404	Not Found	ressource inexistante
405	Method Not Allowed	pas implémenté
409	Conflict	POST ?

Valeurs

- JSON de préférence ! (vs texte, XML...)
 - listes vs dictionnaires vs ...
- compatible JS



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

JSON – STD 90 / RFC 8259 (2017)

- serialisation d'un objet sous forme de texte unicode
- JSON : objet, tableau, valeur num, chaîne, true false null
Python : `dict list int/float str True False None`
- échappement avec *backslash*
- **mais** pas de commentaires...

```
{  
  "id": 16,  
  "nom": "PostgreSQL",  
  "version": 16.1,  
  "liste": [ "hello world!\n", "Calvin\tHobbes\n" ],  
  "updated": false,  
  "none": null,  
  "tags": { "one": 1, "two": 2.0 }  
}
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Région 01

```
GET /regions?code=01 HTTP/1.1
Host: geo.api.gouv.fr
```

Département abc

```
GET /departements/abc HTTP/1.1
Host: geo.api.gouv.fr
```

Réponse JSON

```
HTTP/1.1 200 OK
Server: nginx/1.10.3 (Ubuntu)
Content-Type: application/json; charset=utf-8
Content-Length: 34
```

```
[{"nom": "Guadeloupe", "code": "01"}]
```

Réponse (bof)

```
HTTP/1.1 404 Not Found
Server: nginx/1.10.3 (Ubuntu)
Content-Type: text/plain; charset=utf-8
Content-Length: 9
```

Not Found



Back-end

FC/CM

```
curl -s "https://api-adresse.data.gouv.fr/reverse/?lat=48.845&lon=2.339" | \  
jq .features[0].properties.label
```

```
GET /reverse/?lat=48.845&lon=2.339 HTTP/1.1
```

```
Host: api-adresse.data.gouv.fr
```

```
HTTP/1.1 200 OK
```

```
Server: nginx/1.23.3
```

```
Date: Wed, 10 Jan 2024 08:15:30 GMT
```

```
Content-Type: application/json; charset=utf-8
```

```
Content-Length: 2585
```

```
Connection: keep-alive
```

```
Vary: Origin
```

```
ETag: W/"a19-s7ifV//ge9CBj7Euq+QViFCDCw4"
```

```
X-Cache-Status: MISS
```

```
Access-Control-Allow-Headers: X-Requested-With,Content-Type
```

```
{  
  "label": "60b Boulevard Saint-Michel 75006 Paris", ...  
}
```

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement



Philosophie

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Cycle de vie des données – CRUD(S)

C Create	INSERT
R Read	SELECT
U Update	UPDATE
D Delete	DELETE
S Search	SELECT

Interfaces

- opérations élémentaires CRUD
- opérations de plus haut niveau
- un peu les deux !?

souple, logique côté client
plus rigide, logique côté serveur



Conseils

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Règles sur la partie chemin

caractères ASCII, minuscules, - **pas** _

style court, pluriel ok, **pas** d'extensions ni de types

valeurs uniquement pour identifier une ressource (clé, plutôt primaire)

hiérarchie avec /, **pas** à la fin

Oui /eleves

Non /Liste_de_tous_les_élèves.json/



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

- suppose un modèle de données bien défini !
- utile pour l'implémentation *back-end*
- utile pour le développeur *front-end*

Interface REST

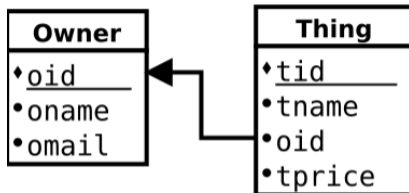
- chemin
- méthodes pertinentes
- paramètres requis/possibles
- sémantique. . .
- gestion des droits
- retours attendus
 - fonctionnement normal
 - **et** gestion des erreurs

préfix et paramètres intégrés
pas toutes !
types, forme. . .
une phrase
qui peut le faire ?
status, json



Exercice

Interface REST de type CRUD



Owner propriétaire

oid clé primaire

oname nom de la personne, unique

omail adresse de messagerie

Thing chose

tid clé primaire

tname nom de la chose

oid clé étrangère vers le propriétaire

tprice valeur de l'objet

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement



7 différences

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Web

vs

REST

méthodes GET/POST

GET/POST/PUT/PATCH/DELETE

user-agent navigateur

application

format document HTML

données JSON

auth page login/logout

par requête, persistent

session serveur

application

http grosses requêtes

petites requêtes nombreuses

back-end plus complexe (templates, sessions)

plus simple



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

<https://www.postman.com/explore>

- nombreuses API, y compris publiques
- problèmes de sécurité : limitation du flux de requêtes
- authentifications. . .
- ne suivent pas souvent les règles !
- Postman : spec, doc, tests. . .



Conseils

- privilégier le format JSON, paramètres et retours
- HTTP coûte cher en latence : agréger les transferts !
- actions asynchrones ? file d'attentes à traiter. . .



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Python DB API



DB API

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

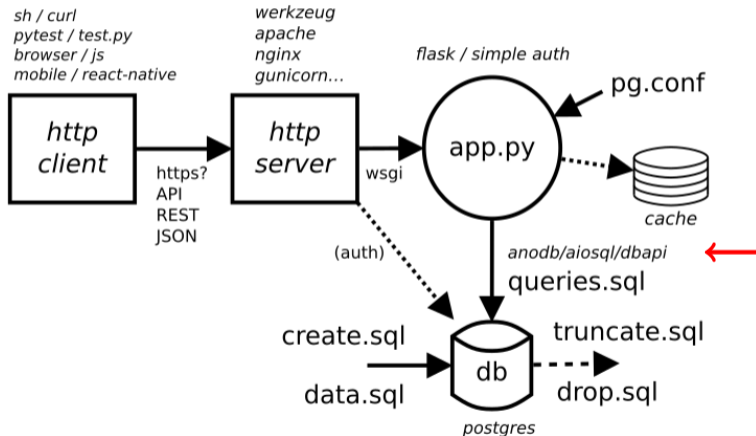
Conseils

CRUDS

AAA

Blueprint

Déploiement





Connexion Python – DB

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Modèle similaire dans tous les langages

- identification de la base de données
- établissement d'une connexion authentifiée
- exécution de divers types de requêtes
- passage de paramètres vs *SQL injection*
- conversion types langages et SQL
- consultation des métadonnées
- gestion des erreurs. . .

Python DB API, Java JDBC, Perl DBI, C ODBC, R DBI, Rust SQLx. . .



Connexion Python – DB

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Cycle de vie

- 1 établissement d'une connexion
- 2 création d'un ou plusieurs curseurs
- 3 exécution d'une requête
- 4 récupération des résultats...
- 5 fermetures...

```
import psycopg as db
conn = db.connect("")
curs = conn.cursor()
curs.execute("SELECT 1, 'un' UNION SELECT 2, 'deux'")
for i in range(curs.rowcount):
    print(curs.fetchone())
curs.close()
conn.close()
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

NON !

- parties facultatives de la spécifications. . .
- variantes de passages de paramètres %s vs ?. . .
- extensions diverses et variées
- chargement explicite du package, pas d'abstraction (*driver*)

```
# NE FONCTIONNE PAS
import sqlite3 as db
conn = db.connect(":memory:")
curs = conn.cursor()
curs.execute("SELECT 1, 'un' UNION SELECT 2, 'deux'")
for i in range(curs.rowcount):
    print(curs.fetchone())
curs.close()
conn.close()
```



Types de connexions et d'implémentations

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

- connexion directe vs réseau
- implémentation du protocole vs utilisation d'une librairie

SQLite

`sqlite3` connexion directe via librairie

Postgres

`psycopg` connexion réseau via librairie `libpq`

`pg8000` connexion réseau via TCP/IP

`pygresql` connexion réseau via librairie `libpq`

`aiopg` version asynchrone au dessus de `psycopg2`



Connexion

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
conn = db.connect(...)
```

- identification de la base de donnée
- authentification...
- options diverses
- support des transactions commit rollback
- cher ! partager si possible, persistance...

```
import psycopg as db
conn = db.connect("host=pagode dbname=comics")
conn = db.connect(host="pagode", dbname="comics", user="calvin", password="5secret")
conn = db.connect(...)
conn.commit()
conn.close()
```



Curseur

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
curs = conn.cursor()
```

- fonctionne avec un contexte **with**
- exécution d'une requête ou d'une commande `execute`
- parcours automatique **for ... in ...**
- parcours manuel `fetchone` `fetchmany` `fetchall`
retour de tuples, **None** si fini
- fermeture `close`

```
# version portable...  
with conn.cursor() as curs:  
    curs.execute("SELECT * FROM Auteur")  
    for row in curs:  
        print(row)
```



Curseur – Retour dictionnaire

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

- résultats par défaut : liste de tuples
- modification avec `row_factory` : dict, objets...

```
import psycopg2 as pg
with pg.connect(row_factory=pg.rows.DictRow) as conn:
    with conn.cursor() as curs:
        curs.execute("SELECT i AS i, i^3 AS i³ FROM generate_series(2, 8) AS i")
        for row in curs:
            print("row:", row)
    conn.commit()
# row: {'i': 2, 'i³': 8.0}
# row: {'i': 3, 'i³': 27.0}
# row: {'i': 4, 'i³': 64.0}
# row: {'i': 5, 'i³': 125.0}
# row: {'i': 6, 'i³': 216.0}
# row: {'i': 7, 'i³': 343.0}
# row: {'i': 8, 'i³': 512.0}
```



Curseur – Passage de paramètres

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

5 styles *qmark numeric format named pyformat*

paramètres tuple ou dictionnaire

portabilité faible, aucun n'est imposé...

merci PEP249 !

psycopg pyformat format

sqlite3 qmark numeric

```
curs.execute("SELECT * FROM Auteur WHERE nom LIKE ?", ("A%",)) # qmark
curs.execute("SELECT * FROM Auteur WHERE nom LIKE :1", ("A%",)) # numeric
curs.execute("SELECT * FROM Auteur WHERE nom LIKE %s", ("A%",)) # format
curs.execute("SELECT * FROM Auteur WHERE nom LIKE :pat", { "pat":"A%" }) # named
curs.execute("SELECT * FROM Auteur WHERE nom LIKE %(pat)s", { "pat":"A%" }) # pyformat
```



Injection de SQL

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Cause erreur d'échappement de valeurs fournies par l'utilisateur

Outil sqlmap analyse et exploitation (e.g. à l'aveugle)...

NE JAMAIS FAIRE ÇA !

```
curs.execute("SELECT * FROM Friend WHERE login='"+ who + "'")
curs.execute("SELECT * FROM Friend WHERE login='%s' % who)
curs.execute("SELECT * FROM Friend WHERE login='%s'.format(who))
curs.execute(f"SELECT * FROM Friend WHERE login='{who}'")
```

mais plutôt ça

```
curs.execute("SELECT * FROM Friend WHERE login=%s", (who, ))
```

Démonstration !

<https://xkcd.com/327/>

- accès aux données... aux méta-données... modification des données...



Requêtes dynamiques

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Composition de SQL en évitant les injections

- driver spécifique, par exemple avec `psycopg.sql`
- échappements selon l'utilisation

`Identifier` identifiant *table, colonne*

`Literal` valeur *string, int, float*

`Placeholder` paramètre de requête à fournir

```
from psycopg import sql
def update_something(conn, tab, col, val):
    with conn.cursor() as curs:
        query = "UPDATE {tab} SET {col} = {val}".format(
            tab=sql.Identifier(tab), col=sql.Identifier(col),
            val=sql.Placeholder("val"))
        curs.execute(query, {"val": val})
```



Conclusion

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

DB API

- connexion Python - DB relationnelles
- compléments et extensions : copy, 2PC, BLOB...
- standard sans être portable !

Abstractions de plus haut niveau

YeSQL encapsulation

AioSQL AnoDB

- cache l'interaction SQL/Python dans des fonctions
- très simple, connaissance et puissance de SQL, plus statique...

ORM *Object Relational Mapper*

SQLAlchemy Django PeeWee

- cache SQL dans une syntaxe Python (DDL, DML...)
- plus complexe, portable, SQL parfois moyen, un peu plus lent...



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

AnoDB



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

- connexion/déconnexion à la base de données
- définition de fonctions par requêtes `-- name: foo`
passage de paramètres (valeurs) nommés `:bla :nom :id :key :val`
- basé sur AioSQL et DB API 2.0, support SQLite Postgres MySQL MariaDB
- cursor disponible pour prendre la main si nécessaire

```
from anodb import DB
db = DB("sqlite3", "things.db", "things.sql")

print("things:", db.get_all_things())           # [(1, "Watch"), (2, "Table"), ...]
print("thing #42:", db.get_a_thing(tid=42))    # ("Car", "Dad")
print("#things:", db.count_things())           # 5432

# remove all things...
db.rm_all_things()
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Requêtes *fixées*

paramètres :param pour des valeurs

	SELECT	liste de tuples
^	SELECT	un seul tuple
\$	SELECT	une seule valeur
!	INSERT UPDATE DELETE (sans RETURNING)	nombre de lignes

```
-- name: get_all_things
```

```
SELECT tid, tname FROM Thing ORDER BY 1;
```

```
-- name: get_a_thing^
```

```
SELECT tname, oname FROM Thing JOIN Owner USING (oid) WHERE tid = :tid;
```

```
-- name: count_things$
```

```
SELECT COUNT(*) AS cnt FROM Thing;
```

```
-- name: rm_all_things!
```

```
DELETE FROM Thing WHERE TRUE;
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Flask et FlaskSimpleAuth



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

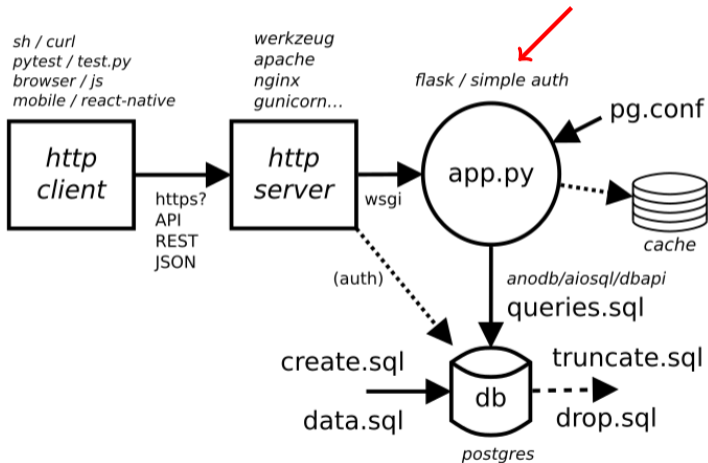
Conseils

CRUDS

AAA

Blueprint

Déploiement





Frameworks Python REST ou Web

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

popularité décroissante

téléchargements par mois, novembre 2023

Flask	micro	106 M/mo
Tornado	micro	31 M/mo
FastAPI	micro async	23 M/mo
Django	web, ORM	14 M/mo
Pyramid	web	3.1 M/mo
Bottle	micro	2.3 M/mo
Sanic	micro	0.8 M/mo
Falcon	micro	0.6 M/mo



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Flask

framework WSGI

- configuration de l'application via chargement de code python
- routage : méthode + chemin → fonction
- récupération des paramètres (HTTP/JSON, chemin)
- génération de réponses JSON, HTML (via template) ; status HTTP
- *event hooks* : avant, après une requête

FlaskSimpleAuth

extension Flask

- gestion automatique des paramètres typés (HTTP/JSON, chemin)
- authentification très configurable
- autorisations explicites sur les routes
- utilitaires : caches, références...



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

- initialisation de la classe
- divers options de configuration
- décorateurs méthode/route → fonction
- autorisations

Flask

app.config

route get post

authorize

```
from FlaskSimpleAuth import Flask
app = Flask("demo")
app.config.from_envvar("APP_CONFIG") # fichier de conf EXTERNE

@app.get("/some/path", authorize="OPEN")
def get_some_path():
    return {"msg": "some path"}, 200

@app.get("/other/path", authorize="OPEN")
def get_other_path():
    return {"hello": "other path"}, 200
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

- tronçons typés `string` `int` `float` `path` `uuid`... name i
- paramètres obligatoires si pas de valeur par défaut j
- paramètres facultatifs si valeur par défaut k
- plusieurs méthodes ? Si même comportement...

```
@app.post("/users/<name>", authorize="OPEN")
def post_users_name(name: str):
    return {"name": name, "msg": f"bonjour {name} !"}, 201

@app.get("/add/<i>", authorize="OPEN")
def get_add_i(i: int, j: int, k: int = 0):
    return str(i+j+k), 200
```




Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

- résultat et statut HTTP
- fonction utile jsonify/json
json *automatique* pour dict...
- voir aussi la classe Response

```
from FlaskSimpleAuth import jsonify, Response

@app.get("/msg/<p>", authorize="OPEN")
def get_msg(p: path):
    return jsonify(f"hello {p} !"), 200

@app.route("/bad", methods=["BAD"], authorize="OPEN")
def bad_bad():
    return Response("bad bad bad!", status=500)
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

- enregistrement de fonctions exécutées
- avant une requête
- ou après

before_request

after_request

```
import logging
log = logging.getLogger("app")

def audit_trail(res: Response) -> Response:
    log.info(f"result code {res.code}")
    return res

app.after_request(audit_trail)
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Authentification

directives FSA_*

FSA_AUTH none httpd basic digest param token...

FSA_REALM authentication realm

FSA_TOKEN_* directives pour l'authentification par token

FSA_PASSWORD_* password management option (algo, params)

```
# exemple de configuration pour FlaskSimpleAuth
```

```
FSA_AUTH = "basic"
```

```
FSA_REALM = "comics"
```

```
FSA_TOKEN_CARRIER = "bearer"
```

```
import logging
```

```
FSA_LOGGING_LEVEL = logging.DEBUG
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Param Authentication

login mdp en paramètres

```
DELETE /users/hobbes HTTP/1.1
Host: www.comics.net
Content-Type: application/json
Content-Length: 33

{"USER":"calvin","PASS":"hobbes"}
```

Basic Authentication

login mdp en base64

```
DELETE /users/hobbes HTTP/1.1
Host: www.comics.net
Authorization: Basic Y2FsZm1uOmhvYmJlcw==
```

Bearer Authentication

token signé

```
DELETE /users/hobbes HTTP/1.1
Host: www.comics.net
Authorization: Bearer comics:calvin:20380119031407:1b098331931e6059
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Pour des tests

- commande principale `flask`
- application via option `--app=...`
- sous-commandes `run routes shell`
- autres options ... `--host=...`
- parfois variables d'environnement
- arrêt brutal `kill flask`

fichier de configuration via l'environnement

```
export APP_CONFIG="./app.conf"
```

démarrage du processus avec redirection des traces dans "app.log"

```
flask --debug --app="app.py" run --host="0.0.0.0" >> app.log 2>&1 &
```

dont on garde le numéro du processus pour envoyer des signaux

```
echo $! > app.pid
```



Tests avec pytest

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Tests internes

- objet `test_client()` de Flask
- méthodes `get` `post` `put` `patch` `delete`...
- entêtes un peu à la main...

Tests externes

- lancer le serveur et lui envoyer des requêtes !
- utiliser l'objet `Session` du module `requests`
- permet de tester un serveur déployé

Authentification ?



Tests internes

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Fichier test_interne.py

■ utilisation de app.test_client()

```
import pytest
from app import app

@pytest.fixture
def client():
    with app.test_client() as c:
        yield c

def test_stuff(client):
    r = client.post("/stuff", data={"stuff": "Book"})
    assert r.status_code == 201
    n = r.json["sid"]
    r = client.get("/stuff")
    assert r.status_code == 200 and b'"Book"' in r.data
    r = client.delete(f"/stuff/{n}")
    assert r.status_code == 204
```



Tests externes

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Fichier test_externe.py

■ utilisation du module requests

```
from requests import Session
requests = Session() # use persistant connections!

import os
URL = os.environ["APP_URL"]

def check_api(method: str, path: str, status: int, **kwargs):
    r = requests.request(method, URL + path, **kwargs)
    assert r.status_code == status
    return r

def test_stuff():
    r = check_api("POST", "/stuff", 201, data={"stuff": "Table"})
    n = r.json["sid"]
    assert r"Table" in check_api("GET", "/stuff", 200).text
    check_api("DELETE", f"/stuff/{n}", 204)
```




Tests mixtes

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Fichier test_mix.py

- utilisation du module FlaskTester, env FLASK_TESTER_APP
 - si URL : tests externes `http://localhost:5000`
 - si package python : tests internes `app`
- gestion de l'authentification, des cookies...

```
import pytest
from FlaskTester import ft_client, ft_authenticator

def test_stuff(ft_client):
    res = ft_client.post("/stuff", 201, data={"stuff": "Book"})
    sid = res.json["sid"]
    ft_client.get("/stuff", 200, b'"Book"')
    ft_client.delete(f"/stuff/{sid}", 204)
```



Design de tests pour une API REST

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Propriétés

systematiques combinaisons méthodes \times routes

complets code *et* résultat attendus

échecs vérifier les erreurs ! oublis, typage. . .

droits authentification, autorisations. . .

indépendants création/destruction des données de tests

nettoyage possible des données de tests

rapides sinon rarement exécutés. . .



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Application client-serveur sur réseau...

- RTFM! `https://xkcd.com/293/`
- activation du mode debug `--debug`
messages verbeux, traces et console dans navigateur...
- sorties de la commande flask `tail -f app.log`
- requêtes manuelles `curl -si -X GET URL...`
- extensions REST pour navigateurs
`Firefox` *RESTer*
`Chrome` *Advanced REST client, Postman*
`Safari` *RESTed*
- tester ! `mypy black/flake8 pytest`



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Conseils



Conseils

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Conception

DB, API

- besoins réels vs imaginaires
- modélisation des données, données de test
- API REST (y compris erreurs), AAA

Répartition des tâches

client vs serveur

- client : tri pour l'affichage
- serveur : requête déterministes

Séparation API et logique applicative

- fonctions et classes pour les aspects *métiers*
- routes focalisées sur les aspects API : HTTP, JSON



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Tests systématiques

- méthodes, chemins, paramètres. . .
- 2xx, 4xx (droits !)
- pas de 5xx en fonctionnement normal ?

Performance

- génération de données, scenarii
- différents niveaux : app, API, DB
dénormalisation, factorisation, cache, index. . .



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

CRUDS



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Fichiers applicatifs

Python, config, SQL

`app.py` implémentation de l'API REST

`database.py` gestion de la base de données

`model.py` définition des types

`app.conf` configuration de l'application (Python)
spécifique à l'instance lancée, chargée par Flask

`create.sql` création du schéma la base de données

`data.sql` données initiales (pour les tests par exemple)

`truncate.sql` efface les données

`drop.sql` efface les tables

`queries.sql` requêtes pour AnoDB



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Modules python

`flasksimpleauth` extension flask

`anodb psycopg` base de données Postgres

`passlib bcrypt` authentification par mot de passe

`pytest flasktester` tests automatiques

```
# create a flask venv
```

```
python -m venv venv
```

```
source venv/bin/activate
```

```
pip install \
```

```
flasksimpleauth anodb psycopg passlib bcrypt \
```

```
flasktester pytest
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Administration

- initialisation du schéma de la base de données

```
createdb stuff
```

```
psql -1 -f create.sql -f data.sql stuff
```

- nettoyage (version très rapide...)

```
dropdb stuff
```

```
CREATE TABLE IF NOT EXISTS
```

```
Stuff(sid SERIAL PRIMARY KEY, stuff TEXT UNIQUE NOT NULL);
```

create.sql

```
INSERT INTO Stuff(stuff) VALUES ('Chair'), ('Desk');
```

data.sql

```
TRUNCATE Stuff;
```

truncate.sql

```
DROP TABLE IF EXISTS Stuff;
```

drop.sql



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Contenu

Python

- variables de l'application : initialisation, connexion...

```
import psycopg
ANODB = { # connection AnoDB / AioSQL / Psycopg
    "db": "psycopg",
    "conn": "dbname=stuff application_name=stuff-backend",
    "queries": "queries.sql",
    "row_factory": psycopg.rows.dict_row,
}

# FlaskSimpleAuth
FSA_MODE = "prod"
FSA_ERROR_RESPONSE = "json:error"
FSA_AUTH = "basic"
FSA_DEFAULT_CONTENT_TYPE = "application/json"

import logging
FSA_LOGGING_LEVEL = logging.DEBUG
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Initialisations

- importations et création de l'application app
- chargement de la configuration via l'environnement APP_CONFIG
- initialisation des modules init_app

```
import logging
logging.basicConfig()
log = logging.getLogger("stuff")
log.setLevel(logging.DEBUG)

from FlaskSimpleAuth import Flask, jsonify, CurrentUser
app = Flask("stuff")
app.config.from_envvar("APP_CONFIG")

import database
database.init_app(app)
from database import db
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Connexion base de données

- utilisation de AnoDB et Reference db
- commit automatique via un *hook* db_commit

```
import FlaskSimpleAuth as fsa # type: ignore
import anodb # type: ignore

db = fsa.Reference()

def db_commit(res: fsa.Response) -> fsa.Response:
    if res.status_code < 400:
        db.commit()
    else: # 4xx user errors, 5xx server errors
        db.rollback()
    return res

def init_app(app: fsa.Flask):
    db.set(anodb.DB(**app.config["ANODB"]))
    app.after_request(db_commit)
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Fonctionnement de Flask

- configuration via environnement et fichier
- commande flask pour tests locaux
démarrage/arrêt du processus, serveur werkzeug

start.sh

```
# fichier de configuration via l'environnement  
export APP_CONFIG="./app.conf"  
# démarrage du processus avec redirection des traces dans "app.log"  
flask --debug --app="app.py" run --host="0.0.0.0" >> app.log 2>&1 &  
# dont on garde le numéro du processus pour envoyer des signaux  
echo $! > app.pid
```

stop.sh

```
# arrêt du processus flask  
kill $(cat app.pid)  
rm -f app.pid
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

GET /version

- teste le fonctionnement de base : HTTP, requête SQL

queries.sql

```
-- name: now$  
SELECT CURRENT_TIMESTAMP;
```

app.py

```
@app.get("/version", authorize="OPEN")  
def get_version():  
    # NOTE json automatique pour les dictionnaires  
    return {"app": app.name, "user": app.current_user(), "now": db.now()}, 200
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

récupération de la version

```
curl -si -X GET http://0.0.0.0:5000/version
```

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json
```

```
Content-Length: 66
```

```
{"app": "stuff", "now": "Wed, 01 May 2024 07:37:36 GMT", "user": null}
```

test

résultat



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

GET /stuff

- liste des trucs, éventuellement filtrée

queries.sql

```
-- name: get_stuff_all
SELECT * FROM Stuff ORDER BY 1;

-- name: get_stuff_like
SELECT * FROM Stuff WHERE stuff LIKE :filter ORDER BY 1;
```

app.py

```
@app.get("/stuff", authorize="OPEN")
def get_stuff(filter: str|None = None):
    if filter:
        res = db.get_stuff_like(filter=filter)
    else:
        res = db.get_stuff_all()
    return jsonify(res), 200
```



Exemple CRUDS

test S

Back-end

FC/CM

```
# tous les trucs
curl -si -X GET http://0.0.0.0:5000/stuff
```

test

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked
```

résultat

```
[{"sid": 1, "stuff": "Chair"}, {"sid": 3, "stuff": "Bed"}]
```

```
# tous les trucs, mais avec un filtre
curl -si -X GET -d filter=% http://0.0.0.0:5000/stuff
```

test

```
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked
```

résultat

```
[{"sid": 1, "stuff": "Chair"}, {"sid": 3, "stuff": "Bed"}]
```



Exemple CRUDS

test S

Back-end

FC/CM

```
# les trucs qui commencent pas C  
curl -si -X GET -d filter=C% http://0.0.0.0:5000/stuff
```

test

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
HTTP/1.1 200 OK  
Content-Type: application/json  
Transfer-Encoding: chunked
```

```
[{"sid": 1, "stuff": "Chair"}]
```

résultat

```
# les trucs qui commencent par A  
curl -si -X GET -d filter=A% http://0.0.0.0:5000/stuff
```

test

```
HTTP/1.1 200 OK  
Content-Type: application/json  
Transfer-Encoding: chunked
```

```
[]
```

résultat



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
GET /stuff/<sid>
```

- récupération d'un truc, retour 200 ou 404

queries.sql

```
-- name: get_stuff_sid^  
SELECT * FROM Stuff WHERE sid = :sid;
```

app.py

```
@app.get("/stuff/<sid>", authorize="OPEN")  
def get_stuff_sid(sid: int):  
    res = db.get_stuff_sid(sid=sid)  
    return (res, 200) if res else ("", 404)
```



Exemple CRUDS

test R

Back-end

FC/CM

test

```
# le truc 1  
curl -si -X GET http://0.0.0.0:5000/stuff/1
```

résultat

```
HTTP/1.1 200 OK  
Content-Type: application/json  
Content-Length: 26
```

```
{"sid":1,"stuff":"Chair"}
```

test

```
# le truc 5432, qui n'existe pas !  
curl -si -X GET http://0.0.0.0:5000/stuff/5432
```

résultat

```
HTTP/1.1 404 NOT FOUND  
Content-Type: application/json  
Content-Length: 0
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

POST /stuff

- création d'un truc, 201 ou 400

queries.sql

```
-- name: add_stuff$  
INSERT INTO Stuff(stuff) VALUES(:stuff) RETURNING sid;
```

app.py

```
@app.post("/stuff", authorize="OPEN")  
def post_stuff(stuff: str):  
    sid = db.add_stuff(stuff=stuff)  
    return {"sid": sid}, 201
```



Exemple CRUDS

test C

Back-end

FC/CM

ajout d'une horloge

```
curl -si -X POST -d stuff=Clock http://0.0.0.0:5000/stuff
```

test

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
HTTP/1.1 201 CREATED
```

```
Content-Type: application/json
```

```
Content-Length: 10
```

```
{"sid":5}
```

résultat

il manque le paramètre "stuff"

```
curl -si -X POST http://0.0.0.0:5000/stuff
```

test

```
HTTP/1.1 400 BAD REQUEST
```

```
Content-Type: application/json
```

```
Content-Length: 43
```

```
{"error": "parameter \"stuff\" is missing"}
```

résultat



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

test

```
# attention, il y a déjà une horloge !  
# erreur interne 500 sur vérification des contraintes...  
# laisser ? détecter à l'avance et retour 409 Conflict ?  
curl -si -X POST -d stuff=Clock http://0.0.0.0:5000/stuff
```

résultat

```
HTTP/1.1 500 INTERNAL SERVER ERROR
```

```
Content-Type: application/json
```

```
Content-Length: 58
```

```
{"error": "internal error caught at parameters on /stuff"}
```




Exemple CRUDS

D

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
DELETE /stuff/<sid>
```

- destruction d'un truc, 204 ou 404

queries.sql

```
-- name: check_stuff_sid^  
SELECT sid FROM stuff WHERE sid = :sid FOR UPDATE;  
  
-- name: del_stuff_sid!  
DELETE FROM Stuff WHERE sid = :sid;
```

app.py

```
@app.delete("/stuff/<sid>", authorize="OPEN")  
def delete_stuff_sid(sid: int):  
    res = db.check_stuff_sid(sid=sid)  
    if not res:  
        return "", 404  
    db.del_stuff_sid(sid=sid)  
    return "", 204
```



Exemple CRUDS

test D

Back-end

FC/CM

```
# efface le truc 2  
curl -si -X DELETE http://0.0.0.0:5000/stuff/2
```

test

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
HTTP/1.1 404 NOT FOUND  
Content-Type: application/json  
Content-Length: 0
```

résultat

```
# plus de truc 2...  
curl -si -X GET http://0.0.0.0:5000/stuff/2
```

test

```
HTTP/1.1 404 NOT FOUND  
Content-Type: application/json  
Content-Length: 0
```

résultat



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

attention, déjà effacé

```
curl -si -X DELETE http://0.0.0.0:5000/stuff/2
```

```
HTTP/1.1 404 NOT FOUND
```

```
Content-Type: application/json
```

```
Content-Length: 0
```

test

résultat



Exemple CRUDS

U

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
PATCH /stuff/<sid> ou PUT /stuff/<sid>
```

- modification d'un truc, 204 ou 400 ou 404

queries.sql

```
-- name: upd_stuff_sid!  
UPDATE Stuff SET stuff = :stuff WHERE sid = :sid;
```

app.py

```
@app.route("/stuff/<sid>", methods=["PUT", "PATCH"], authorize="OPEN")  
def patch_stuff(sid: int, stuff: str):  
    res = db.check_stuff_sid(sid=sid)  
    if not res:  
        return "", 404  
    db.upd_stuff_sid(sid=sid, stuff=stuff)  
    return "", 204
```



Exemple CRUDS

test U

Back-end

FC/CM

```
# modification du truc 3  
curl -si -X PUT -d stuff=Bed http://0.0.0.0:5000/stuff/3
```

test

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
HTTP/1.1 204 NO CONTENT  
Content-Type: application/json
```

résultat

```
# le truc 3 a bien été modifié...  
curl -si -X GET http://0.0.0.0:5000/stuff/3
```

test

```
HTTP/1.1 200 OK  
Content-Type: application/json  
Content-Length: 24
```

résultat

```
{"sid":3,"stuff":"Bed"}
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

modification du truc 5432, mais il n'existe pas...

```
curl -si -X PATCH -d stuff=Wardrobe http://0.0.0.0:5000/stuff/5432
```

HTTP/1.1 404 NOT FOUND

Content-Type: application/json

Content-Length: 0

test

résultat



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

dataclass, pydantic...

- définition de structures de données (types)
- éventuellement avec des contraintes de validation

```
import pydantic

class User(pydantic.BaseModel):
    login: str
    password: str
    isAdmin: bool

    # contrainte sur la longueur du mot de passe
    @pydantic.field_validator('password')
    @classmethod
    def min_len(cls, v: str) -> str:
        if len(v) < 4:
            raise ValueError("password too short!")
        return v
```

model.py



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

queries.sql

```
-- name: new_user$  
INSERT INTO Utilisateur(login, pword, isAdmin)  
VALUES (:login, :pw, :adm) ON CONFLICT DO NOTHING RETURNING id;
```

```
import model  
  
@app.post("/users-mgt", authorize="OPEN") # MUST NOT BE OPEN!  
def post_users_mgt(u: model.User):  
    uid = db.new_user(login=u.login, pw=u.password, adm=u.isAdmin)  
    if uid is None:  
        return "", 409  
    return jsonify(uid), 201
```

app.py



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
curl -si -X POST \  
  -H "Content-Type: application/json" \  
  -d '{"u":{"login":"bro","password":"bro-pass","isAdmin":false}}' \  
  http://0.0.0.0:5000/users-mgt
```

test

```
HTTP/1.1 409 CONFLICT  
Content-Type: application/json  
Content-Length: 0
```

résultat



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

AAA

Authentication – Authorization – Audit



Exemple AAA

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

AAA

Authentication vérification identité QUI ?

- serveur web ou framework ou application...
- *login/mdp, token, certificat...*
- coût de la vérification...

Authorization droit de faire une opération QUOI ?

- rôles dans l'application – *groupes*
- logique applicative – *données de l'utilisateur*

Audit génération de traces QUAND ?

- serveur web
- application WSGI
- base de données...



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Sécurité déclarative

- modes d'authentification : `basic token...`
- authorizations : conditions requises sur une route
 - `auth` authentification simple
 - `groupes` applicatifs
 - `oauth` délégation des droits, par opérations
 - `perms` permissions liées aux objets et opérations
- code applicatif plus simple, conditions garanties



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Authentication avec FlaskSimpleAuth

mode	none httpd fake basic param digest token
passer <i>h(s.p)</i>	sel s, mdp <i>p</i>
	fonctions conçues pour être coûteuses... <i>400 ms</i>

create.sql

```
CREATE TABLE IF NOT EXISTS Utilisateur(
  id SERIAL PRIMARY KEY,
  login TEXT UNIQUE NOT NULL,
  pword TEXT NOT NULL,
  isAdmin BOOL NOT NULL DEFAULT FALSE,
  created TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

queries.sql

```
-- name: user_perms^
SELECT pword, isAdmin FROM Utilisateur WHERE login = :login;
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Fonctions de support de FlaskSimpleAuth

- mot de passe ou `None` get_user_pass
- appartenance à un groupe group_check user_in_group

app.py

```
@app.get_user_pass
def get_user_pass(user):
    res = db.user_perms(login=user)
    return res["pword"] if res else None

@app.group_check("ADMIN")
def user_is_admin(user):
    res = db.user_perms(login=user)
    return res["isadmin"] if res else False
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Stockage des mots de passe

- module python passlib *plaintext bcrypt ldap...*
- fonction de hash bcrypt avec 2^4 rounds *quelques ms*

pass2csv.py

```
import re
import fileinput
import passlib.context as pc

pm = pc.CryptContext("bcrypt", bcrypt__default_rounds=4) # password manager

for line in fileinput.input():
    if not re.match(r"^\s*(#|$)", line): # skip comments and blank lines
        w = re.split(r"[ \t]", line.strip(), 2)
        print(f'"{w[0]}", "{pm.hash(w[1])}", {w[2] if len(w) > 2 else None}')
```



Exemple AAA

chargement mdp

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

users.in

```
# login pass raw,list,of, csv,stuff
calvin hobbles TRUE
hobbles susie FALSE
susie derkins FALSE
```

shell

```
python ./pass2csv.py < users.in > users.csv
```

users.csv

```
"calvin", "$2b$04$sgNm3EE0yqyQNUDCV2aloeM.xpfhDEnvKOG1JPkgq6jHm4TwGeUpC", TRUE
"hobbles", "$2b$04$Qq1WxaJ03QwxgtRuj5IrXuNjFkHqk5GEZknIdMQzq9ZVZA0JRL.ba", FALSE
"susie", "$2b$04$0/nWZnmTvS52GGGAaDJaRuWhJi4/7qtnEEgqS6tw7VdkAsYw8iHgy", FALSE
```

load.sql

```
Utilisateur(login, pword, isAdmin) from './users.csv' (format csv)
```




Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Autorisation via paramètre `authorize` du décorateur `route`

CLOSE valeur par défaut, 403 !

OPEN aucune authentification requise

AUTH tous les utilisateurs authentifiés

... groupes applicatifs

`app.py`

```
@app.get("/hello", authorize="AUTH")
def get_hello():
    user = app.get_user()
    return jsonify(f"hello {user}!"), 200

@app.get("/admin", authorize="ADMIN")
def get_admin(user: CurrentUser):
    return jsonify(f"hello admin {user}!"), 200
```



Exemple AAA

test hello

Back-end

FC/CM

```
# pas d'authentification  
curl -si -X GET http://0.0.0.0:5000/hello
```

test

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
HTTP/1.1 401 UNAUTHORIZED  
Content-Type: application/json  
Content-Length: 41  
WWW-Authenticate: Basic realm="stuff", Bearer realm="stuff"
```

résultat

```
{"error": "missing authorization header"}
```

```
# utilisateur inexistant  
curl -si -X GET -u hacker:secret http://0.0.0.0:5000/hello
```

test

```
HTTP/1.1 401 UNAUTHORIZED  
Content-Type: application/json  
Content-Length: 33  
WWW-Authenticate: Basic realm="stuff", Bearer realm="stuff"
```

résultat

```
{"error": "no such user: hacker"}
```



Exemple AAA

test hello

Back-end

FC/CM

```
# mauvais mot de passe
```

```
curl -si -X GET -u hobbes:pas-le-bon http://0.0.0.0:5000/hello
```

test

résultat

```
HTTP/1.1 401 UNAUTHORIZED
```

```
Content-Type: application/json
```

```
Content-Length: 40
```

```
WWW-Authenticate: Basic realm="stuff", Bearer realm="stuff"
```

```
{"error": "invalid password for hobbes"}
```

```
# ok, hobbes est un utilisateur
```

```
curl -si -X GET -u hobbes:susie http://0.0.0.0:5000/hello
```

test

résultat

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json
```

```
Content-Length: 16
```

```
"hello hobbes!"
```



Exemple AAA

test admin

Back-end

FC/CM

```
# non, hobbes n'est pas admin  
curl -si -X GET -u hobbes:susie http://0.0.0.0:5000/admin
```

test

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
HTTP/1.1 403 FORBIDDEN  
Content-Type: application/json  
Content-Length: 35
```

résultat

```
{"error": "not in group \"ADMIN\""}

---


```

```
# ok, calvin est admin  
curl -si -X GET -u calvin:hobbes http://0.0.0.0:5000/admin
```

test

```
HTTP/1.1 200 OK  
Content-Type: application/json  
Content-Length: 22
```

résultat

```
"hello admin calvin!"
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Enregistrement d'un nouvel utilisateur

- route ouverte à tous, mot de passe, droits par défaut...
- vérification éventuelle ?

queries.sql

```
-- name: add_new_user!  
INSERT INTO Utilisateur(login, pword) VALUES (:login, :pword);
```

app.py

```
@app.post("/register", authorize="OPEN")  
def post_register(login: str, pword: str):  
    # FIXME check whether user already exists...  
    db.add_new_user(login=login, pword=app.hash_password(pword))  
    return "", 201
```



Exemple AAA

test register

Back-end

FC/CM

```
# add new user "moe" with password "secret"  
curl -si -X POST -d login=moe -d pword=secret http://0.0.0.0:5000/register
```

test

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
HTTP/1.1 500 INTERNAL SERVER ERROR  
Content-Type: application/json  
Content-Length: 61
```

résultat

```
{"error": "internal error caught at parameters on /register"}
```

```
# calvin already exists  
curl -si -X POST -d login=calvin -d pword=comics http://0.0.0.0:5000/register
```

test

```
HTTP/1.1 500 INTERNAL SERVER ERROR  
Content-Type: application/json  
Content-Length: 61
```

résultat

```
{"error": "internal error caught at parameters on /register"}
```



Exemple AAA

whoami

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

app.py

```
# affiche l'utilisateur authentifié
@app.get("/whoami", authorize="AUTH")
def get_whoami(user: CurrentUser):
    return jsonify(user), 200
```

test

```
# route acceptant une authentification "token" ou "basic"
curl -si -X GET -u moe:secret http://0.0.0.0:5000/whoami
```

résultat

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 6

"moe"
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Authentification par token

- beaucoup moins coûteux qu'une vérification de mot de passe
connexion avec mdp → token, puis utilisation jusqu'à expiration
- format FSA realm:user:timestamp:signature
exemple *kiva:calvin:20380119031407:bdc82cef86b08aa9607fd16e6a03985f*
- standard JWT RFC 7519
- transport : *bearer, cookie, paramètre...*

app.py

```
# création d'un token pour l'utilisateur authentifié
@app.get("/login", authorize="AUTH", auth="basic")
def get_login(user: CurrentUser):
    return jsonify(app.create_token(user)), 200
```




Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

test

récupération d'un token...

```
curl -si -X GET -u moe:secret http://0.0.0.0:5000/login
```

résultat

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: 44

"stuff:moe:20240501083736:5802d6e6151ecb8c"



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
curl -si -X GET \  
  -H "Authorization: Bearer stuff:moe:20240501083736:5802d6e6151ecb8c" \  
  http://0.0.0.0:5000/whoami
```

```
HTTP/1.1 200 OK  
Content-Type: application/json  
Content-Length: 6
```

```
"moe"
```

test

résultat



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

test

```
curl -si -X GET \  
  -H "Authorization: Bearer stuff:moe:30240501083736:5802d6e6151ecb8c" \  
  http://0.0.0.0:5000/whoami
```

résultat

```
HTTP/1.1 401 UNAUTHORIZED  
Content-Type: application/json  
Content-Length: 44  
WWW-Authenticate: Basic realm="stuff", Bearer realm="stuff"  
  
{"error": "unexpected Authorization header"}
```



Exemple AAA

qui suis-je ?

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

app.py

```
# utilisateur authentifié par token uniquement
@app.get("/qui-suis-je", authorize="AUTH", auth="token")
def get_qui_suis_je(user: CurrentUser):
    return jsonify(user), 200
```

test

```
# cette route requiert une authentification "token"
curl -si -X GET -u moe:secret http://0.0.0.0:5000/qui-suis-je
```

résultat

```
HTTP/1.1 401 UNAUTHORIZED
Content-Type: application/json
Content-Length: 26
WWW-Authenticate: Bearer realm="stuff"

{"error": "missing token"}
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
curl -si -X GET \  
  -H "Authorization: Bearer stuff:moe:20240501083736:5802d6e6151ecb8c" \  
  http://0.0.0.0:5000/qui-suis-je
```

```
HTTP/1.1 200 OK  
Content-Type: application/json  
Content-Length: 6
```

```
"moe"
```

test

résultat



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Objets avec un propriétaire

- restriction d'accès aux propriétaires des objets

create.sql

```
CREATE TABLE IF NOT EXISTS Owned(  
  oid SERIAL PRIMARY KEY,  
  object TEXT NOT NULL,  
  owner INTEGER NOT NULL REFERENCES Utilisateur,  
  UNIQUE(object, owner)  
);  
  
CREATE INDEX IF NOT EXISTS owned_owner ON Owned(owner);
```



Exemple AAA

permissions

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

app.py

```
@app.get("/owned", authorize="ADMIN")
def get_owned():
    res = db.get_owned()
    return jsonify(res), 200

@app.get("/object", authorize="ADMIN")
def get_object():
    res = db.get_owned()
    return jsonify(res), 200
```

```
-- name: get_owned
SELECT oid, object, owner
FROM Owned
ORDER BY 1;
```

queries.sql



Exemple AAA

permissions

Back-end

FC/CM

```
# calvin is an admin  
curl -si -X GET -u calvin:hobbes http://0.0.0.0:5000/owned
```

test

```
HTTP/1.1 200 OK  
Content-Type: application/json  
Transfer-Encoding: chunked
```

résultat

```
[{"oid": 1, "object": "Box", "owner": 1}, {"oid": 2, "object": "Box", "owner": 2}, {"oid": 3, "object": "Box", "owner": 3}]
```

Conseils

```
# susie is not an admin  
curl -si -X GET -u susie:derkins http://0.0.0.0:5000/owned
```

test

```
HTTP/1.1 403 FORBIDDEN  
Content-Type: application/json  
Content-Length: 35
```

résultat

```
{"error": "not in group \"ADMIN\""} 
```

Architecture

REST

DB API

AnoDB

Flask

CRUDS

AAA

Blueprint

Déploiement



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
# calvin is an admin
```

```
curl -si -X GET -u calvin:hobbes http://0.0.0.0:5000/object
```

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json
```

```
Transfer-Encoding: chunked
```

```
[{"oid": 1, "object": "Box", "owner": 1}, {"oid": 2, "object": "Box", "owner": 2}, {"oid": 3, "object": "Box", "owner": 3}
```

test

résultat



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Enregistrement de permissions

- l'utilisatrice *login* peut-elle accéder à l'object *oid* pour l'opération *mode* ?
- **True** *Ok* **False** *403 Forbidden* **None** *404 Not Found*

app.py

```
# a user can access an object they own
@app.object_perms("owned")
def check_owned_perms(login: str, oid: int, _mode):
    res = db.can_access_owned(login=login, oid=oid)
    return res["owned"] if res else None
```

queries.sql

```
-- name: can_access_owned~
SELECT u.login = :login AS owned
FROM Owned AS o JOIN Utilisateur AS u ON (o.owner = u.id)
WHERE o.oid = :oid;
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Accès à un objet particulier

- dont on est propriétaire...

app.py

```
@app.get("/object/<oid>", authorize=("owned", "oid"))
def get_object_oid(oid: int):
    res = db.get_object_oid(oid=oid)
    return res, 200
```

queries.sql

```
-- name: get_object_oid~
SELECT oid, object
FROM Owned
WHERE oid = :oid;
```



Exemple AAA

permissions

Back-end

FC/CM

```
# calvin can see his Box
```

```
curl -si -X GET -u calvin:hobbes http://0.0.0.0:5000/object/1
```

test

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json
```

```
Content-Length: 25
```

```
{"object": "Box", "oid": 1}
```

résultat

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
# hobbes cannot see calvin's Box
```

```
curl -si -X GET -u hobbes:susie http://0.0.0.0:5000/object/1
```

test

```
HTTP/1.1 403 FORBIDDEN
```

```
Content-Type: application/json
```

```
Content-Length: 48
```

```
{"error": "permission denied on owned:1 (None)"}
```

résultat



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Enregistrement de permissions

- l'utilisateur *login* peut-il accéder à l'utilisateur *uid* pour l'opération *mode* ?

app.py

```
# a user can access data related to them
@app.object_perms("user")
def check_user_perms(login: str, uid: int, _mode):
    res = db.can_access_user(login=login, uid=uid)
    return res["self"] if res else None
```

queries.sql

```
-- name: can_access_user^
SELECT u.login = :login AS self
FROM Utilisateur AS u
WHERE u.id = :uid;
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Accès aux objets d'un utilisateur

- soi-même...

app.py

```
@app.get("/owned/<uid>", authorize=("user",))
def get_owned_uid(uid: int):
    res = db.get_owned_uid(uid=uid)
    # NOTE jsonify pas indispensable...
    return jsonify(res), 200
```

queries.sql

```
-- name: get_owned_uid
SELECT oid, object, owner
FROM Owned
WHERE owner = :uid;
```



Exemple AAA

permissions

Back-end

FC/CM

```
# calvin can see his own objects  
curl -si -X GET -u calvin:hobbes http://0.0.0.0:5000/owned/1
```

test

```
HTTP/1.1 200 OK  
Content-Type: application/json  
Transfer-Encoding: chunked
```

résultat

```
[{"oid": 1, "object": "Box", "owner": 1}, {"oid": 4, "object": "Tiger", "owner": 1}, {"oid": 5,
```

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
# hobbes cannot access calvin's objects  
curl -si -X GET -u hobbes:susie http://0.0.0.0:5000/owned/1
```

test

```
HTTP/1.1 403 FORBIDDEN  
Content-Type: application/json  
Content-Length: 47
```

résultat

```
{"error": "permission denied on user:1 (None)"}
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

test

```
# there is no user 42
```

```
curl -si -X GET -u hobbes:susie http://0.0.0.0:5000/owned/42
```

résultat

```
HTTP/1.1 404 NOT FOUND
```

```
Content-Type: application/json
```

```
Content-Length: 29
```

```
{"error": "object not found"}
```




Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

AAA avec FlaskSimpleAuth

Simple à mettre en place !

Authentification utiliser basic et token sur TLS

- forcer un login et l'utilisation de token
- durée de vie des tokens selon application ?
- renouvellement ? multi-facteur ?

Authorisations modèle déclaratif et complet

`authorize`

- modèle par rôle très vite limité
- permissions par objets. . .

Performances cache avec TTL automatique

Extensions possibles : *recovery code*



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Blueprint



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Blueprint

- organisation en plusieurs fichiers d'une application flask
- enregistrement différé des routes dans l'application

```
from compute import comp
app.register_blueprint(comp, url_prefix="/cmp")
```

app.py



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Authentification avec FlaskSimpleAuth

- attention au partage de variables `current_app` db

```
from flask import Blueprint, current_app as app, jsonify
comp = Blueprint("compute", __name__)
```

```
@comp.get("/add", authorize="AUTH")
```

```
def get_add(i: int, j: int):
    return jsonify(i+j), 200
```

```
@comp.get("/hash", authorize="AUTH")
```

```
def get_hash(apass: str):
    return jsonify(app.hash_password(apass)), 200
```

compute.py



Exemple Blueprint

tests

Back-end

FC/CM

```
# gestion de paramètres
```

```
curl -si -X GET -u moe:secret -d i=30 -d j=12 http://0.0.0.0:5000/cmp/add
```

test

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json
```

```
Content-Length: 3
```

```
42
```

résultat

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
# gestion de paramètres
```

```
curl -si -X GET -u moe:secret -d apass="Wow!" http://0.0.0.0:5000/cmp/hash
```

test

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json
```

```
Content-Length: 63
```

```
"$2y$04$G24zuVu7mFGRMGyKNm/KQ.rpsFJGz1r3jJgNiGb1WPZG3Zo0b3L4u"
```

résultat



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Déploiement

(Mise en production)



Cycle de vie d'un *back-end*

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Development

- tests en *local* (si possible) : machines de dev assez puissantes
- tests automatiques à distance : *Continuous Integration/Delivery* (CI/CD)

Production

- mise en ligne du service, base avec données initiales, caches. . .
- éventuellement environnement de *pre-production*

Maintenance et sécurité

- corrections de bugs, nouveaux services, modifications des données. . .
- problématique : gestion des versions, des secrets. . .

Fin de vie (Retirement)

- transfert/archivage des données ?



Déploiement d'une application WSGI

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

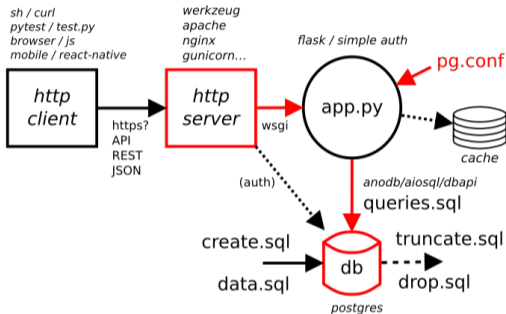
Conseils

CRUDS

AAA

Blueprint

Déploiement



base de données configuration, initialisation, droits

application fichiers, droits, accès DB...

serveur web/WSGI site, interface WSGI, droits...



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Compte et base de données applicative

psql

- utilisateur kiva, base de donnée kiva, droits ?

```
CREATE ROLE kiva WITH
  LOGIN ENCRYPTED PASSWORD 'EfTLPJBijALPZT0tuk8nAo'
  CONNECTION LIMIT 3
  USER susie, calvin, hobbes, moe;
```

```
CREATE DATABASE kiva WITH
  OWNER kiva
  CONNECTION LIMIT 5;
```

```
\c kiva
ALTER DEFAULT PRIVILEGES IN SCHEMA PUBLIC
  GRANT ALL PRIVILEGES ON TABLES TO pg_database_owner;
ALTER DEFAULT PRIVILEGES IN SCHEMA PUBLIC
  GRANT ALL PRIVILEGES ON ROUTINES TO pg_database_owner;
ALTER DEFAULT PRIVILEGES IN SCHEMA PUBLIC
  GRANT ALL PRIVILEGES ON SEQUENCES TO pg_database_owner;
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Accès à la base de données

pg_hba.conf

- autorisation d'accès à partir du serveur
- chiffrement ? impact sur les performances ?

```
# access to database kiva with role kiva from wsgi server with password authn  
hostssl kiva kiva 10.101.1.100/32 scram-sha-256  
# group access with identd authn  
hostssl kiva +kiva 10.201.8.88/32 ident
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Compte utilisateur

kiva@mobapp-srv

- compte utilisateur avec accès par SSH
- répertoire(s) de l'application
- authentification pour la base de données

```
sudo adduser --disabled-password kiva
# ~kiva/.ssh/authorized_keys: ...
# ~kiva/.pgpass: pagode:5432:kiva:kiva:EfTLPJBijAlPZT0tuk8nAo
# ~kiva/app: répertoire de l'application
# mais aussi : conf static www venv...
```

Transfert des fichiers *dev* vers (*pre-*)*prod*

rsync

```
rsync -av *.py *.sql ... kiva@mobapp.minesparis.psl.eu:app/
```

Création des données initiales (une seule fois en *prod* !)

psql

```
psql -1 -f drop.sql -f create.sql -f data.sql "host=pagode user=kiva dbname=kiva"
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Lancement de l'application

app/kiva.wsgi

```
# app configuration
from os import environ
environ["APP_NAME"] = "kiva"
environ["APP_CONFIG"] = "/home/kiva/conf/server.conf"
environ["APP_SECRET"] = "..."

# import flask application: app.py / app + application
from app import app as application
```

Configuration de l'application

conf/server.conf

```
import psycopg

ANODB = {
    "db": "postgres",
    "conn": "host=pagode user=kiva dbname=kiva application_name=kiva-app",
    "queries": "queries.sql",
    "row_factory": psycopg.rows.dict_row,
}
```



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
WSGIPythonHome "/home/kiva/venv"
WSGIPythonPath "/home/kiva/venv/lib/python3.12/site-packages"
WSGIPythonOptimize 2

<Directory /home/kiva>
    WSGIProcessGroup kiva
    WSGIApplicationGroup kiva
    Require all granted
</Directory>

<VirtualHost *:443>
    ServerName kiva.mobapp.minesparis.psl.eu
    # SSL, logs, document root...

    WSGIDaemonProcess kiva \
        lang=C.UTF-8 locale=C.UTF-8 user=kiva group=kiva \
        processes=1 threads=1 dispname=kiva home="/home/kiva/app"

    WSGIScriptAlias "/api" "/home/kiva/app/kiva.wsgi"
    WSGIPassAuthorization On
</VirtualHost>
```



Déploiement en pratique

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Équipe *prod* (Claire, Fabien, Laurent...)

OS création des comptes bases de données et unix, accès SSH...

Platform conf apache, venv, WSGI, application...

Équipe *dev* (vous)

- déploiement semi-automatique `make deploy`

```
make deploy
```

```
# rsync ssh psql...
```

```
curl -si -X GET https://kiva.mobapp.minesparis.psl.eu/api/version
```

```
# TADA...
```



Au delà...

<https://www.fullstackpython.com/>

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Problématiques

prod/dev

Version de l'application, du schéma, des données...

Performance monitoring, load balancer, caches, async, index, pool...

Sûreté monitoring, (haute) disponibilité, sauvegardes, PCA, PRA...

Sécurité parefeu, logs, surveillance, proxy SSL, secrets...

Maintenance mises à jours (de sécurité), obsolescence OS/applications

Compétences

prod

- installation et maintenance de l'infrastructure matérielle (ou *Cloud*)
accès, réseau, serveurs, baies de disques, alimentation, refroidissement, ...
- installation et administration de machines (ou services)
comptes utilisateurs, gestion des accès, sécurité, sauvegardes, maj, obsolescence, ...
- installation et administration des services (interdépendants) nécessaires
Apache, Python/Flask/..., Postgres, Redis...