

Programmation back-end

Fabien Coelho, Claire Medrala

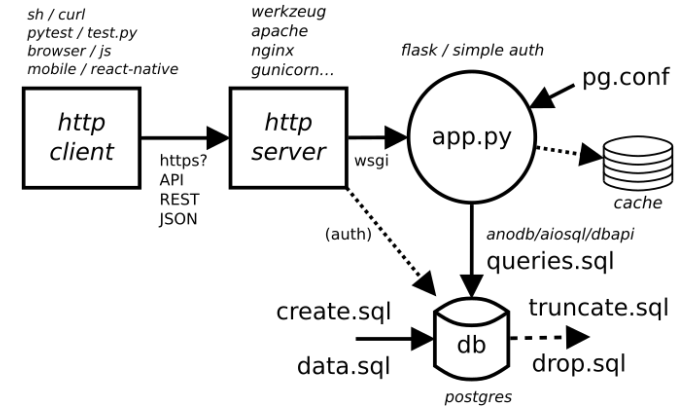
Mines Paris – PSL

Avril 2024

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

Architecture back-end

très simplifiée



Architecture back-end



Architecture back-end

Technologies



Architecture back-end

Sécurité – AAA

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

HTTP	HyperText Transport Protocol	RFC 2616
WSGI	Python Web Server Gateway Interface	PEP 333
ASGI	Python Asynchronous Server Gateway Interface	
Flask	requête HTTP : conf, params, auth, réponses, json. . .	
	<i>app</i> votre application !	
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	

5 / 135

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

6 / 135



Architecture back-end. . .

Exemple AWS



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

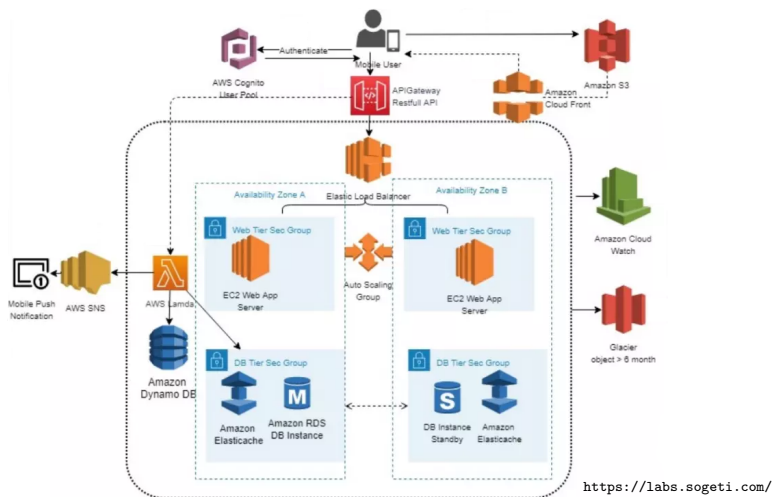
Conseils

CRUDS

AAA

Blueprint

Déploiement



7 / 135

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

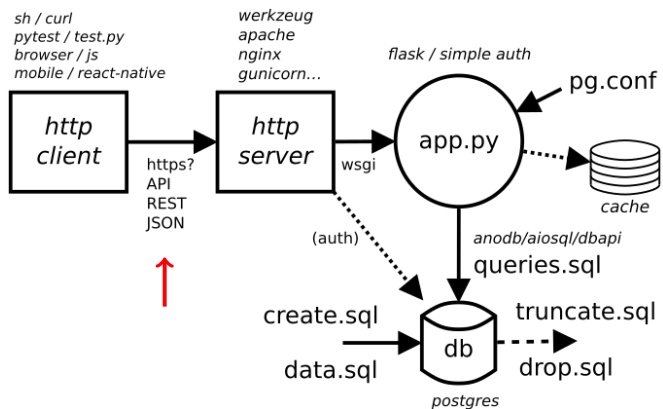
Déploiement

REST

8 / 135



API REST



9 / 135



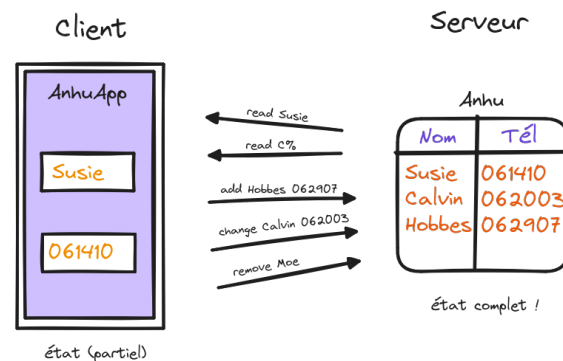
REST – REpresentational State Transfer

Architecture logicielle

Interface

- transfert d'états

identification et manipulation de ressources



10 / 135



REST – REpresentational State Transfer



HTTP – protocole client-serveur

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...

11 / 135

Back-end
FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

`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 Y2Fsdlu0m1kcA==
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

12 / 135



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"}
```

13 / 135



REST / HTTP – conventions

Back-end
FC/CM
Architecture
REST
DB API
AnoDB
Flask
Conseils
CRUDS
AAA
Blueprint
Déploiement

Méthodes

GET	consultation	SELECT
POST	ajout (retour <i>id</i> ?)	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

14 / 135



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

15 / 135



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 Y2Fsdm1uOmlkcA==
Content-Length: 16
Content-Type: application/json

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

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"]]
```

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 dicts

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

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

16 / 135



REST / HTTP – retours

état et valeurs



JSON – format

JavaScript Object Notation

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

17 / 135

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 }
}
```

18 / 135



REST / HTTP – exemples

geo.api.gouv.fr



REST / HTTP – exemples

api-adresse.data.gouv.fr

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
```

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"}]
```

Département abc

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

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

19 / 135

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
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+QViFDCw4"
X-Cache-Status: MISS
Access-Control-Allow-Headers: X-Requested-With,Content-Type

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

20 / 135



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 *souple, logique côté client*
- opérations de plus haut niveau *plus rigide, logique côté serveur*
- un peu les deux !?

21 / 135



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/

22 / 135



Exercice

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 *préfix et paramètres intégrés*
- méthodes pertinentes *pas toutes !*
- paramètres requis/possibles *types, forme... .*
- sémantique... *une phrase*
- gestion des droits *qui peut le faire ?*
- retours attendus *status, json*
 - fonctionnement normal
 - **et** gestion des erreurs

23 / 135



Définition d'une API REST

Back-end
FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

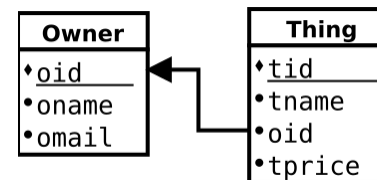
AAA

Blueprint

Déploiement

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

24 / 135



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

25 / 135



Exemples d'API REST

OpenAPI

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. . .

26 / 135



Python DB API

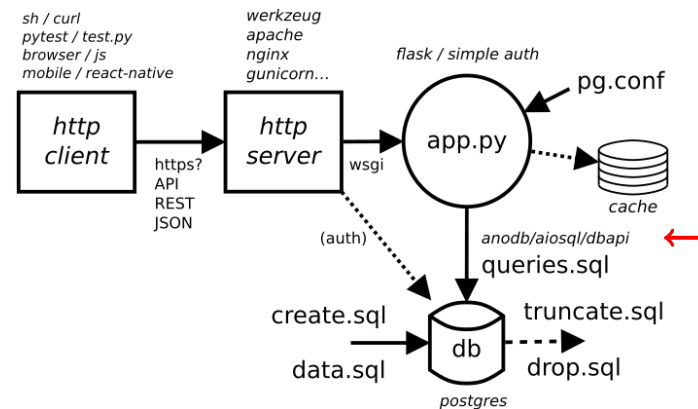
Back-end
FC/CM
Architecture
REST
DB API
AnoDB
Flask
Conseils
CRUDS
AAA
Blueprint
Déploiement

27 / 135



DB API

Back-end
FC/CM
Architecture
REST
DB API
AnoDB
Flask
Conseils
CRUDS
AAA
Blueprint
Déploiement



28 / 135



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...

29 / 135



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()
```

30 / 135



Portabilité ?

DB API 2.0 – PEP249



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()
```

31 / 135

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

32 / 135



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()
```

33 / 135



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

35 / 135



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)
```

34 / 135



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
```

36 / 135



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
 - Identifient identifiant *table, colonne*
 - Littéral 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...



AnoDB



SQL en Python

YeSQL!



SQL en Python

définition des requêtes, suffix

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()
```

41 / 135

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;
```

42 / 135



Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Flask et FlaskSimpleAuth

43 / 135



HTTP en Python

WSGI

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

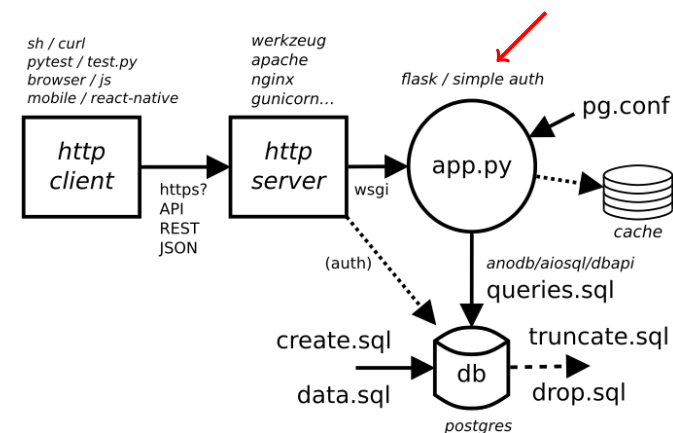
Conseils

CRUDS

AAA

Blueprint

Déploiement



44 / 135



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

45 / 135



HTTP en Python

Flask et FlaskSimpleAuth

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...

46 / 135



HTTP en Python

Usage général

Back-end

FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

- initialisation de la classe Flask
- divers options de configuration app.config
- décorateurs méthode/route → fonction route get post
- autorisations 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
```

47 / 135



HTTP en Python

chemin, méthode, paramètres

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
```

48 / 135



HTTP en Python

Retours



HTTP en Python

Hooks

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)
```

49 / 135

Back-end
FC/CM
Architecture
REST
DB API
AnoDB
Flask
Conseils
CRUDS
AAA
Blueprint
Déploiement

- enregistrement de fonctions exécutées
- avant une requête before_request
- ou après 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)
```

50 / 135



HTTP en Python

Configuration



HTTP en Python

Authentification

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
```

51 / 135

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 Y2FsdmluOmhvYmJlcw==
```

Bearer Authentication token signé

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

52 / 135



HTTP en Python

Démarrage/arrêt



Tests avec pytest

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
```

53 / 135

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 ?

54 / 135



Tests internes



Tests externes

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
```

55 / 135

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 persistent 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)
```

56 / 135



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)
```

57 / 135



Design de tests pour une API REST

Back-end
FC/CM
Architecture
REST
DB API
AnoDB
Flask
Conseils
CRUDS
AAA
Blueprint
Déploiement

Propriétés

- systématiques** combinaisons méthodes × 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...

58 / 135



HTTP en Python

en pratique...

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`

59 / 135



Conseils

60 / 135



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

61 / 135



Conseils

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...

62 / 135



Back-end
FC/CM
Architecture
REST
DB API
AnoDB
Flask
Conseils
CRUDS
AAA
Blueprint
Déploiement

CRUDS

63 / 135



Exemple CRUDS

Fichiers

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

64 / 135



Exemple CRUDS

Python Virtual Environment



Exemple CRUDS

Fichiers SQL

- 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
```

65 / 135



Exemple CRUDS

Fichier *app.conf*



Exemple CRUDS

Fichier *app.py*

- 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

# connection à la base de données (AnoDB / AioSQL / Psycopg)
DB_TYPE = "psycopg"
DB_CONN = "dbname=stuff application_name=stuff-backend"
DB_SQL = "queries.sql"
DB_OPTIONS = { "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
```

67 / 135

- 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

66 / 135

- Back-end
- FC/CM
- Architecture
- REST
- DB API
- AnoDB
- Flask
- Conseils
- CRUDS
- AAA
- Blueprint
- Déploiement

Initialisations

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

```
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
```

68 / 135



Exemple CRUDS

Fichier *database.py*



Exemple CRUDS

start/stop

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):
    cf = app.config
    db.set(anodb.DB(cf["DB_TYPE"], cf["DB_CONN"], cf["DB_SQL"], **cf["DB_OPTIONS"]))
    app.after_request(db_commit)
```

69 / 135

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

```
# 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

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

start.sh

stop.sh

70 / 135



Exemple CRUDS

version



Exemple CRUDS

test version

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

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

@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
```

queries.sql

app.py

71 / 135

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": "Fri, 05 Apr 2024 07:56:27 GMT", "user": null}
```

test

résultat

72 / 135



Exemple CRUDS

S

Back-end
FC/CM

GET /stuff

- liste des trucs, éventuellement filtrée

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

```
-- 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.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
```

queries.sql

app.py

73 / 135



Exemple CRUDS

test S

Back-end
FC/CM

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

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"}, {"sid": 2, "stuff": "Desk"}]
```

test

résultat

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

test

résultat

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

[{"sid": 1, "stuff": "Chair"}, {"sid": 2, "stuff": "Desk"}]
```

74 / 135



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
```

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

75 / 135



Exemple CRUDS

R

Back-end
FC/CM

GET /stuff/<sid>

- récupération d'un truc, retour 200 ou 404

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

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

queries.sql

```
@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)
```

app.py

76 / 135



Exemple CRUDS

test R



Exemple CRUDS

C

Back-end
FC/CM
Architecture
REST
DB API
AnoDB
Flask
Conseils
CRUDS
AAA
Blueprint
Déploiement

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

test

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
```

77 / 135



Exemple CRUDS

test C



Exemple CRUDS

test C

Back-end
FC/CM
Architecture
REST
DB API
AnoDB
Flask
Conseils
CRUDS
AAA
Blueprint
Déploiement

```
# ajout d'une horloge
curl -si -X POST -d stuff=Clock http://0.0.0.0:5000/stuff
```

test

résultat

```
HTTP/1.1 201 CREATED
Content-Type: application/json
Content-Length: 10

{"sid":3}
```

test

```
# il manque le paramètre "stuff"
curl -si -X POST http://0.0.0.0:5000/stuff
```

résultat

```
HTTP/1.1 400 BAD REQUEST
Content-Type: application/json
Content-Length: 43

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

79 / 135

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
```

78 / 135

```
# 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
```

test

résultat

```
HTTP/1.1 500 INTERNAL SERVER ERROR
Content-Type: application/json
Content-Length: 58
```

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

80 / 135



Exemple CRUDS

D



Exemple CRUDS

test 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

```
-- name: check_stuff_sid~
SELECT sid FROM stuff WHERE sid = :sid FOR UPDATE;

-- name: del_stuff_sid!
DELETE FROM Stuff WHERE sid = :sid;

@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
```

queries.sql

app.py

81 / 135

Back-end
FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

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

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

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

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

test

résultat

test

résultat

82 / 135



Exemple CRUDS

test D



Exemple CRUDS

U

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

83 / 135

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

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

@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
```

queries.sql

app.py

84 / 135



Exemple CRUDS

test U



Exemple CRUDS

test U

- Back-end
- FC/CM
- Architecture
- REST
- DB API
- AnoDB
- Flask
- Conseils
- CRUDS
- AAA
- Blueprint
- Déploiement

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

test

résultat

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

test

résultat

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

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

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

85 / 135



Exemple CRUDS

pydantic



Exemple CRUDS

pydantic

- 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

model.py

```
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
```

87 / 135

- 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

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

queries.sql

```
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

88 / 135



Exemple CRUDS

test pydantic / JSON



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/1.1 201 CREATED  
Content-Type: application/json  
Content-Length: 2
```

6

résultat

89 / 135



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...

91 / 135

Back-end
FC/CM
Architecture
REST
DB API
AnoDB
Flask
Conseils
CRUDS
AAA
Blueprint
Déploiement

AAA

Authentication – Authorization – Audit

90 / 135

Exemple AAA

Philosophie

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

92 / 135



Exemple AAA

données



Exemple AAA

callbacks

Back-end
FC/CM

Architecture

REST

DB API

AnoDB

Flask

Conseils

CRUDS

AAA

Blueprint

Déploiement

Authentification avec FlaskSimpleAuth

mode none httpd fake basic param digest token
 passe $h(s.p)$ sel s, mdp p
 fonctions conçues pour être coûteuses... 400 ms

create.sql

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

queries.sql

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

93 / 135

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
```

94 / 135



Exemple AAA

mots de passe



Exemple AAA

chargement mdp

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}')
```

95 / 135

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 hobbes TRUE
hobbes susie FALSE
susie derkins FALSE
```

shell

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

users.csv

```
"calvin", "$2b$04$iXlyigEo/A2AT7Nr3L9k/01KqDEq/Ri4jubolag1Iav6WxN7Rm7am", TRUE
"hobbes", "$2b$04$Jon5Y9QumYXXpWRTHC8byuUJeRD1j17q3fLntZwv/EzImbvUrBhpi", FALSE
"susie", "$2b$04$PK5MCKGXp0XFNEvyELJqHeZ1SWSFVAFNZV0vJusJnsCAEx1kyb91.", FALSE
```

load.sql

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

96 / 135



Exemple AAA

authorization



Exemple AAA

test hello

- 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
```

97 / 135

- Back-end
- FC/CM
- Architecture
- REST
- DB API
- AnoDB
- Flask
- Conseils
- CRUDS
- AAA
- Blueprint
- Déploiement

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

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

{"error": "missing authorization header"}

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

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

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

test

résultat

test

résultat

98 / 135



Exemple AAA

test hello



Exemple AAA

test admin

- Back-end
- FC/CM
- Architecture
- REST
- DB API
- AnoDB
- Flask
- Conseils
- CRUDS
- AAA
- Blueprint
- Déploiement

```
# mauvais mot de passe
curl -si -X GET -u hobbes:pas-le-bon http://0.0.0.0:5000/hello

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

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 16

"hello hobbes!"
```

test

résultat

test

résultat

99 / 135

- Back-end
- FC/CM
- Architecture
- REST
- DB API
- AnoDB
- Flask
- Conseils
- CRUDS
- AAA
- Blueprint
- Déploiement

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

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

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

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

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

"hello admin calvin!"
```

test

résultat

test

résultat

100 / 135



Exemple AAA

register



Exemple AAA

test register

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



Exemple AAA

whoami



Exemple AAA

token

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

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
```

101 / 135

```
# 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

```
HTTP/1.1 201 CREATED
Content-Type: application/json
Content-Length: 0
```

résultat

```
# 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

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

résultat

102 / 135

```
# 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"
```

103 / 135

Authentification par token

- beaucoup moins coûteux qu'une vérification de mot de passe
connection 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
```

104 / 135



Exemple AAA

test token



Exemple AAA

test token valide

- Back-end
- FC/CM
- Architecture
- REST
- DB API
- AnoDB
- Flask
- Conseils
- CRUDS
- AAA
- Blueprint
- Déploiement

```
# récupération d'un token...
curl -si -X GET -u moe:secret http://0.0.0.0:5000/login

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 44

"stuff:moe:20240405085627:cac16b4dee01e31b"
```

test

résultat

105 / 135

- Back-end
- FC/CM
- Architecture
- REST
- DB API
- AnoDB
- Flask
- Conseils
- CRUDS
- AAA
- Blueprint
- Déploiement

```
curl -si -X GET \
-H "Authorization: Bearer stuff:moe:20240405085627:cac16b4dee01e31b" \
http://0.0.0.0:5000/whoami

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

"moe"
```

test

résultat

106 / 135



Exemple AAA

test token invalide



Exemple AAA

qui suis-je ?

- Back-end
- FC/CM
- Architecture
- REST
- DB API
- AnoDB
- Flask
- Conseils
- CRUDS
- AAA
- Blueprint
- Déploiement

```
curl -si -X GET \
-H "Authorization: Bearer stuff:moe:30240405085627:cac16b4dee01e31b" \
http://0.0.0.0:5000/whoami

HTTP/1.1 401 UNAUTHORIZED
Content-Type: application/json
Content-Length: 44
WWW-Authenticate: Basic realm="stuff", Bearer realm="stuff"

{"error": "unexpected Authorization header"}
```

test

résultat

107 / 135

- Back-end
- FC/CM
- Architecture
- REST
- DB API
- AnoDB
- Flask
- Conseils
- CRUDS
- AAA
- Blueprint
- Déploiement

```
# 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

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

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

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

app.py

test

résultat

108 / 135



Exemple AAA

test token



Exemple AAA

permissions

- Back-end
- FC/CM
- Architecture
- REST
- DB API
- AnoDB
- Flask
- Conseils
- CRUDS
- AAA
- Blueprint
- Déploiement

```
curl -si -X GET \
-H "Authorization: Bearer stuff:moe:20240405085627:cac16b4dee01e31b" \
http://0.0.0.0:5000/qui-suis-je
```

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

"moe"

test

résultat

109 / 135

- 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 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);
```

create.sql

110 / 135



Exemple AAA

permissions



Exemple AAA

permissions

- Back-end
- FC/CM
- Architecture
- REST
- DB API
- AnoDB
- Flask
- Conseils
- CRUDS
- AAA
- Blueprint
- Déploiement

```
@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;
```

app.py

queries.sql

111 / 135

- 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/owned
```

```
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}]
```

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

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

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

test

résultat

test

résultat

112 / 135



Exemple AAA

permissions

- 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, "o
```

test

résultat



Exemple AAA

permissions

- 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'objet *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;
```



Exemple AAA

permissions

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

app.py

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

queries.sql



Exemple AAA

permissions

- Back-end
- FC/CM
- Architecture
- REST
- DB API
- AnoDB
- Flask
- Conseils
- CRUDS
- AAA
- Blueprint
- Déploiement

```
# 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
```

résultat

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

test

```
# hobbes cannot see calvin's Box
curl -si -X GET -u hobbes:susie http://0.0.0.0:5000/object/1
```

résultat

```
HTTP/1.1 403 FORBIDDEN
Content-Type: application/json
Content-Length: 48
```

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



Exemple AAA

permissions



Exemple AAA

permissions

- 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* ?

```
# 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
```

app.py

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

queries.sql

117 / 135



Exemple AAA

permissions



Exemple AAA

permissions

- Back-end
- FC/CM
- Architecture
- REST
- DB API
- AnoDB
- Flask
- Conseils
- CRUDS
- AAA
- Blueprint
- Déploiement

```
# 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

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

résultat

```
# 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

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

résultat

119 / 135

- Back-end
- FC/CM
- Architecture
- REST
- DB API
- AnoDB
- Flask
- Conseils
- CRUDS
- AAA
- Blueprint
- Déploiement

Accès aux objets d'un utilisateur

- soi-même...

```
@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
```

app.py

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

queries.sql

118 / 135

```
# there is no user 42
curl -si -X GET -u hobbes:susie http://0.0.0.0:5000/owned/42
```

test

```
HTTP/1.1 404 NOT FOUND
Content-Type: application/json
Content-Length: 29

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

résultat

120 / 135



Exemple AAA

conclusion



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

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

Performances cache avec TTL automatique

Extensions possibles : *recovery code*

authorize

121 / 135

Back-end
FC/CM
Architecture
REST
DB API
AnoDB
Flask
Conseils
CRUDS
AAA
Blueprint
Déploiement

Blueprint

122 / 135



Exemple Blueprint

importation



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

123 / 135

Back-end
FC/CM
Architecture
REST
DB API
AnoDB
Flask
Conseils
CRUDS
AAA
Blueprint
Déploiement

Exemple Blueprint

définition

Authentification avec FlaskSimpleAuth

- attention au partage de variables `current_app db`

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

compute.py

```
@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
```

124 / 135



Exemple Blueprint

tests



Back-end

test

FC/CM

Back-end

Architecture

Architecture

REST

REST

DB API

DB API

AnoDB

AnoDB

Flask

Flask

Conseils

Conseils

CRUDS

CRUDS

AAA

AAA

Blueprint

Blueprint

Déploiement

Déploiement

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

résultat

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 3
```

42

test

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

résultat

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 63
```

```
"$2y$04$JrdIuWsqfB1X49Lk932kHu4w6AXwUJexJfqH0ehWKoz0s526kJ0oy"
```

125 / 135

Déploiement

(Mise en production)

126 / 135



Cycle de vie d'un *back-end*



Déploiement d'une application WSGI

Back-end

Back-end

FC/CM

FC/CM

Architecture

Architecture

REST

REST

DB API

DB API

AnoDB

AnoDB

Flask

Flask

Conseils

Conseils

CRUDS

CRUDS

AAA

AAA

Blueprint

Blueprint

Déploiement

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*

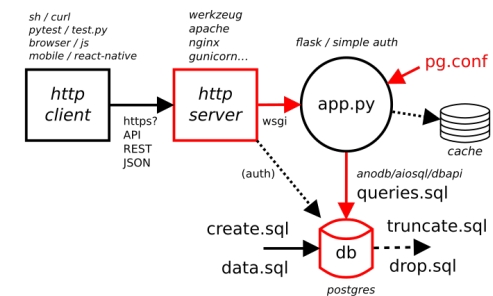
Maintenance et sécurité

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

Fin de vie (Retirement)

- transfert/archivage des données ?

127 / 135



base de données configuration, initialisation, droits
 application fichiers, droits, accès DB. . .
 serveur web/WSGI site, interface WSGI, droits. . .

128 / 135



Postgres

pagode



Postgres

pagode

- 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;
```

129 / 135



Application

mobapp-srv



Application

mobapp-srv

- 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"
```

131 / 135

- 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 usgi server with password authn
hostssl kiva kiva 10.101.1.100/32 scram-sha-256
# group access with ident authn
hostssl kiva +kiva 10.201.8.88/32 ident
```

130 / 135

- 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 pycopg
DB_TYPE = "postgres"
DB_CONN = "host=pagode user=kiva dbname=kiva application_name=kiva-app"
DB_SQL = "queries.sql"
DB_OPTIONS = { "row_factory": pycopg.rows.dict_row }
```

```
FSA_AUTH = "basic"
FSA_TOKEN_SECRET = os.environ["APP_SECRET"]
```

132 / 135



Server web : apache, wsgi, venv

kiva-httpd.conf



Déploiement en pratique

```

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 display-name=kiva home="/home/kiva/app"

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

```

133 / 135

```

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...

```

134 / 135



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...

135 / 135