Ready ?

created by Mehmet Nuri Yumuşak

Complete Backend Development with Python A-Z™

I’m Mehmet Nuri Yumuşak

4 Years in backend development (in startup companies)

Overview of the course

- Backend Architecture
- Building API from scratch
- Complete backend design
- Portable, easy, fast and secure deployment in production

What is backend?

- Connections
- Database
- Main Logic
- Serving
- Core

Why it is important?

If you don’t have a good backend

- Smiling while you are suffering
- Always problem
- Different to change
- Think big

After this course?

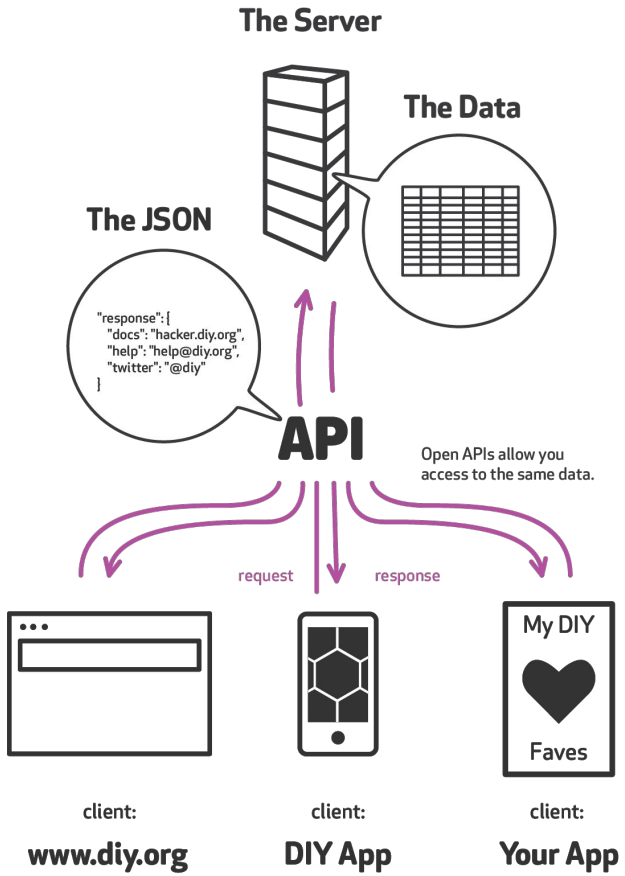
You will be able to;

- Design backend architecture according to needs
- Well documented, fast and secure API
- Scalable system
- System that can be maintained easily

What is API?

Application Programming Interface

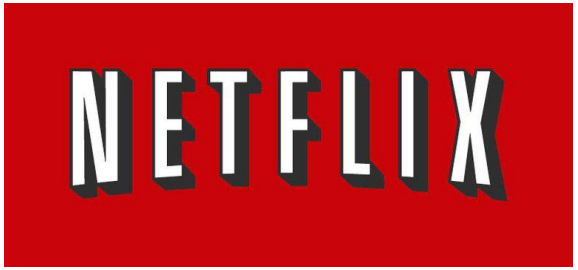
- governs the *access point(s)* for the server.
- communication between services
- database connection



Why python for API?

- Easy Language, Easy Development
- Various Library for any purpose
- Modern and fast (faster than JAVA)
- Less dependency, easy installment
- Good frameworks

Companies use python



Python API Frameworks

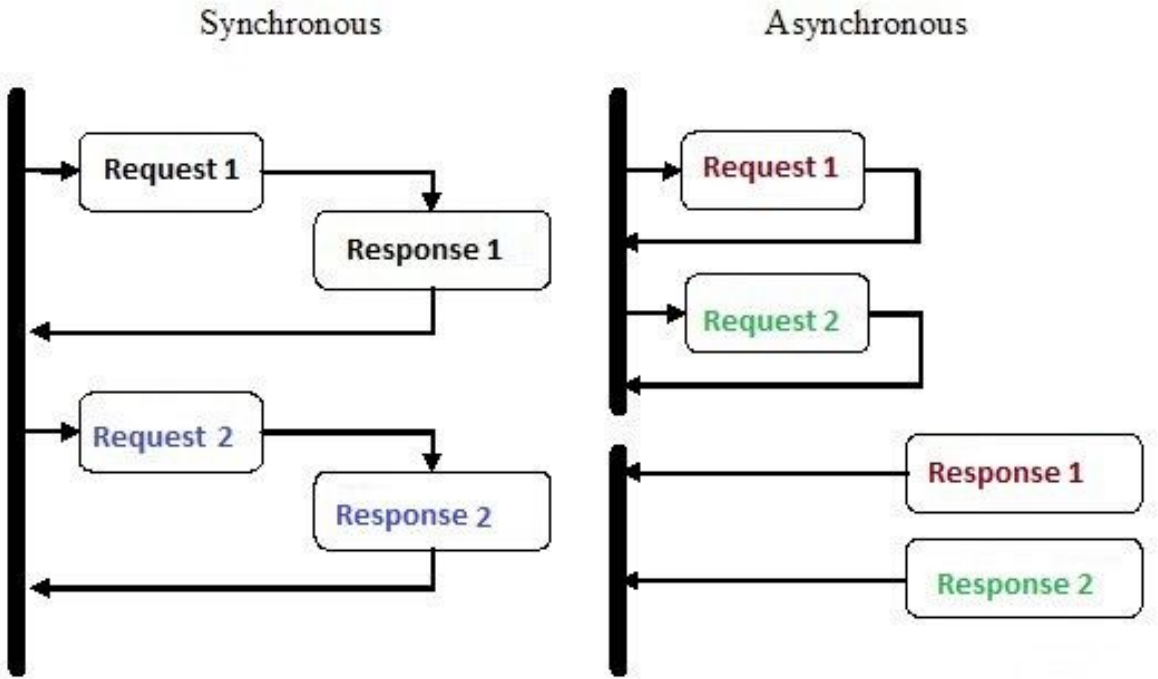
SYNC

- Flask
- Django
- Pyramid

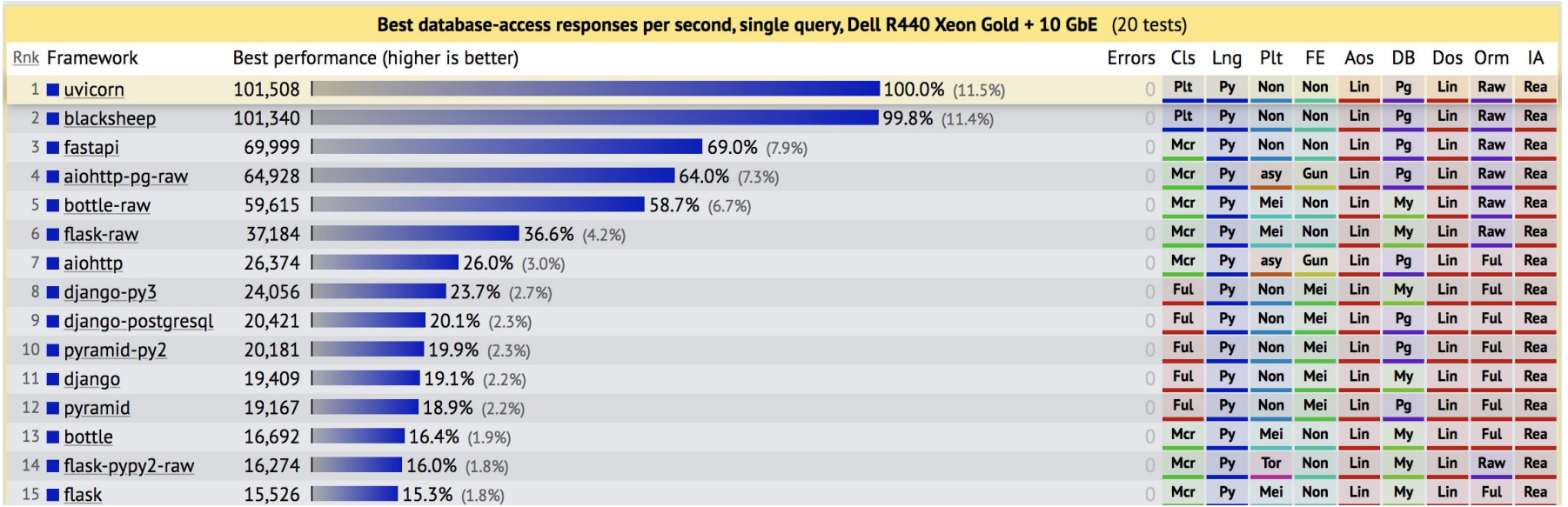
ASYNC

- Tornado
- aiohttp
- FastAPI

Sync vs Async



Benchmarks



Best database-access responses per second, single query, Dell R440 Xeon Gold + 10 GbE (20 tests)

Rnk	Framework	Best performance (higher is better)		Errors	Cls	Lng	Plt	FE	Aos	DB	Dos	Orm	IA
1	uvicorn	101,508	100.0% (11.5%)	0	Plt	Py	Non	Non	Lin	Pg	Lin	Raw	Rea
2	blacksheep	101,340	99.8% (11.4%)	0	Plt	Py	Non	Non	Lin	Pg	Lin	Raw	Rea
3	fastapi	69,999	69.0% (7.9%)	0	Mcr	Py	Non	Non	Lin	Pg	Lin	Raw	Rea
4	aiohttp-pg-raw	64,928	64.0% (7.3%)	0	Mcr	Py	asy	Gun	Lin	Pg	Lin	Raw	Rea
5	bottle-raw	59,615	58.7% (6.7%)	0	Mcr	Py	Mei	Non	Lin	My	Lin	Raw	Rea
6	flask-raw	37,184	36.6% (4.2%)	0	Mcr	Py	Mei	Non	Lin	My	Lin	Raw	Rea
7	aiohttp	26,374	26.0% (3.0%)	0	Mcr	Py	asy	Gun	Lin	Pg	Lin	Ful	Rea
8	django-py3	24,056	23.7% (2.7%)	0	Ful	Py	Non	Mei	Lin	My	Lin	Ful	Rea
9	django-postgresql	20,421	20.1% (2.3%)	0	Ful	Py	Non	Mei	Lin	Pg	Lin	Ful	Rea
10	pyramid-py2	20,181	19.9% (2.3%)	0	Ful	Py	Non	Mei	Lin	Pg	Lin	Ful	Rea
11	django	19,409	19.1% (2.2%)	0	Ful	Py	Non	Mei	Lin	My	Lin	Ful	Rea
12	pyramid	19,167	18.9% (2.2%)	0	Ful	Py	Non	Mei	Lin	Pg	Lin	Ful	Rea
13	bottle	16,692	16.4% (1.9%)	0	Mcr	Py	Mei	Non	Lin	My	Lin	Ful	Rea
14	flask-pypy2-raw	16,274	16.0% (1.8%)	0	Mcr	Py	Tor	Non	Lin	My	Lin	Raw	Rea
15	flask	15,526	15.3% (1.8%)	0	Mcr	Py	Mei	Non	Lin	My	Lin	Ful	Rea

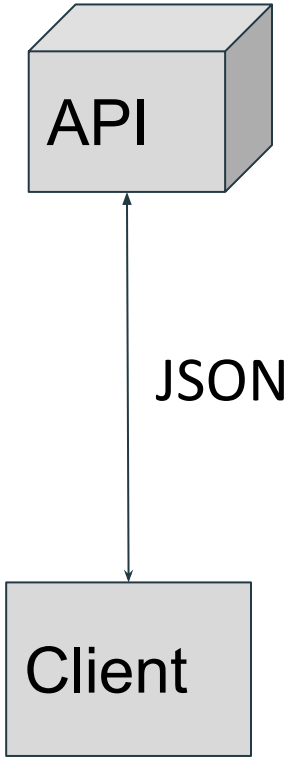
We are going to use fastapi

What is JSON?

Is a language between API and client.

- Easy to understand for human
- Easy to parse for machine
- Support most known data types

```
{
        “object-key-1”: “string value”,
        “object-key-2”: [1, 2, 3],
        “object-key-3”: {
                “sub-object-key-1”: 12.23
        }
}
```



HTTP

Communication between clients and servers is done by HTTP requests and responses

HTTP Requests has;

- Type (GET, POST, UPDATE, DELETE etc.)
- Headers
- Payload (JSON or different)

HTTP

```
POST /cgi-bin/process.cgi HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.tutorialspoint.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive

<?xml version="1.0" encoding="utf-8"?>
<string xmlns="http://clearforest.com/">string</string>
```

HTTP Methods

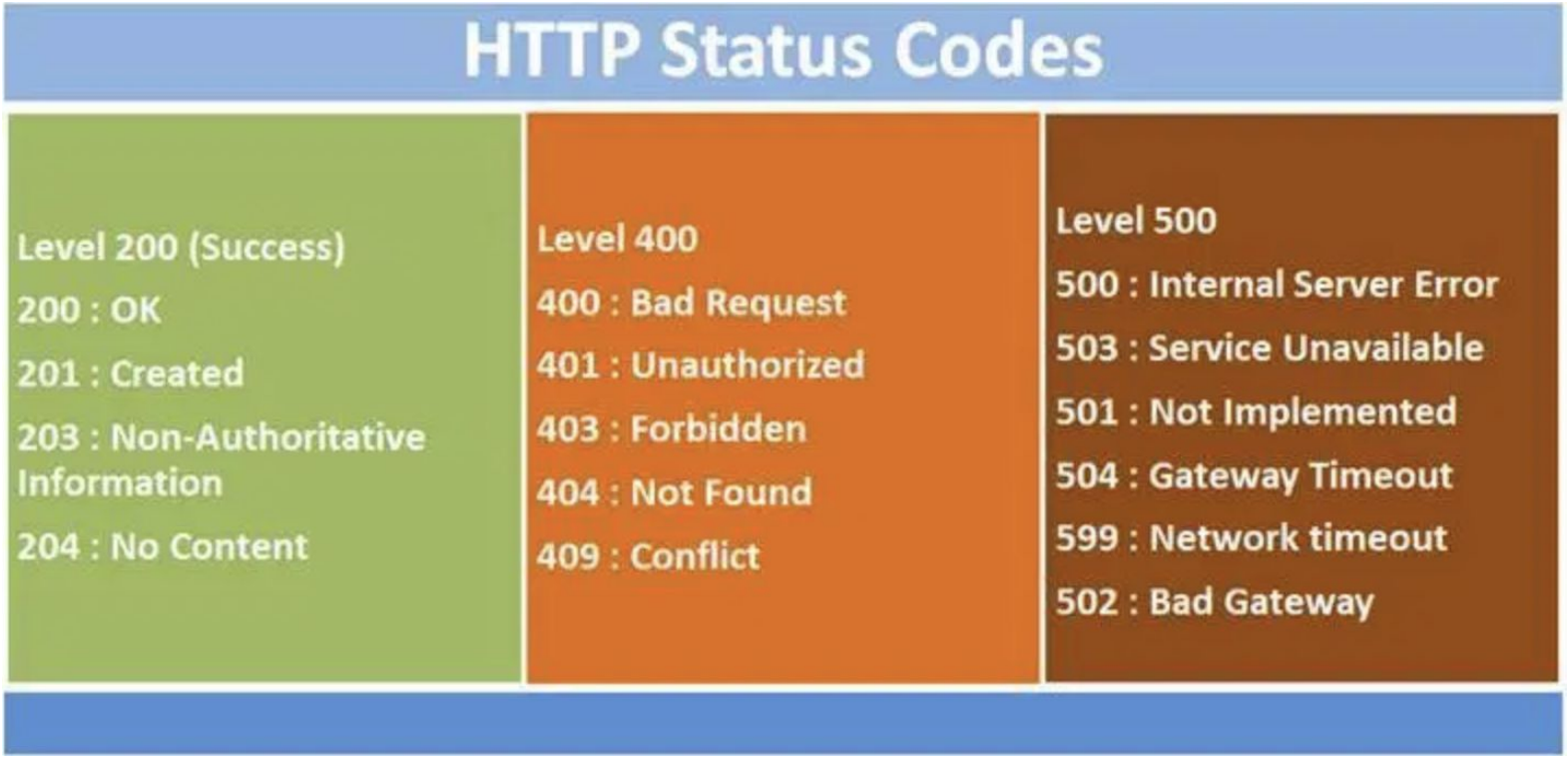
GET	POST	DELETE	UPDATE	PATCH
No payload	Has payload	No payload	Has payload	Has payload
To fetch data	To insert data	To delete data	To update all data	To update some part of data
Doesn’t change database	Change database	Change database	Change database	Change database

HTTP Status Codes

When serves respond to client, HTTP response has a status code.

- 1XX - Informative
- 2XX - Success
- 3XX - Redirection
- 4XX - Client Error
- 5XX - Server Error

HTTP Status Codes



Our Sample Project

It will be a BOOKSTORE API

- Books
- Authors
- Magazines

Naming Endpoints

- Always use **nouns** not verbs
- Could be understood easily
- Follow HTTP Method rules
- Parameters comes after ? and separated with &

Naming Endpoints

WRONG	CORRECT
GET - /getAllBooks	GET - /books
POST - /createBook	POST - /book
GET - /checkLoginUser	GET - /validation/user

Bookstore API Endpoints

For Login System

POST	/user	Save bookstore admin to db
PUT	/user	Update user info
DELETE	/user	Delete the user
GET	/user?password	Check if user exist

Bookstore API Endpoints

For saving items

POST	/book	Save book to db
POST	/magazine	Save magazine to db
POST	/author	Save author to db
PUT	/book	Update book
PUT	/magazine	Update magazine
PATCH	/author/name	Update author's name

Bookstore API Endpoints

For deleting items

DELETE	/book	Delete book from db
DELETE	/magazine	Delete magazine from db
DELETE	/author	Delete author from db

Bookstore API Endpoints

For getting books

GET	/books	Get all books
GET	/book/isbn	Get specific book
GET	/book?category	Get all books in given category
GET	/book?year	Get all books in given publishing year

Bookstore API Endpoints

For getting author

GET	/authors	Get all authors
GET	/author/id	Get info about author
GET	/author/id/book?order&category	Get all books of an given author with order type and category

Versioning

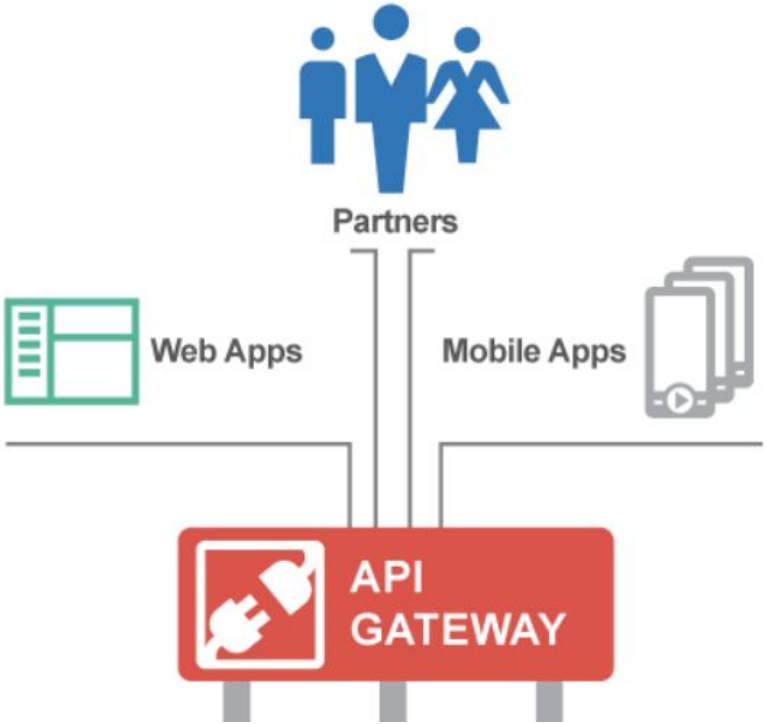
URL Versioning

- http://www.api.com/v1/users
- http://www.api.com/v2/users
- http://www.api.com/v3/users

Header Versioning

- Accept-version: 1
- Accept-version: 2
- Accept-version: 3

Versioning



Authentication vs Authorization

Authentication is when an entity proves an identity. In other words, Authentication proves that you are who you say you are.

Authorization is an entirely different concept and in simple terms, Authorization is when an entity proves a right to access.

Authentication vs Authorization

In summary:

Authentication: Refers to proving correct identity

Authorization: Refers to allowing a certain action

An API might authenticate you but not authorize you to make a certain request.

Authentication Schemas

- Basic Authentication
- API Keys
- **Oauth 2.0 (JWT)**

All of them should be in HTTPS(SSL) network.

Basic Authentication

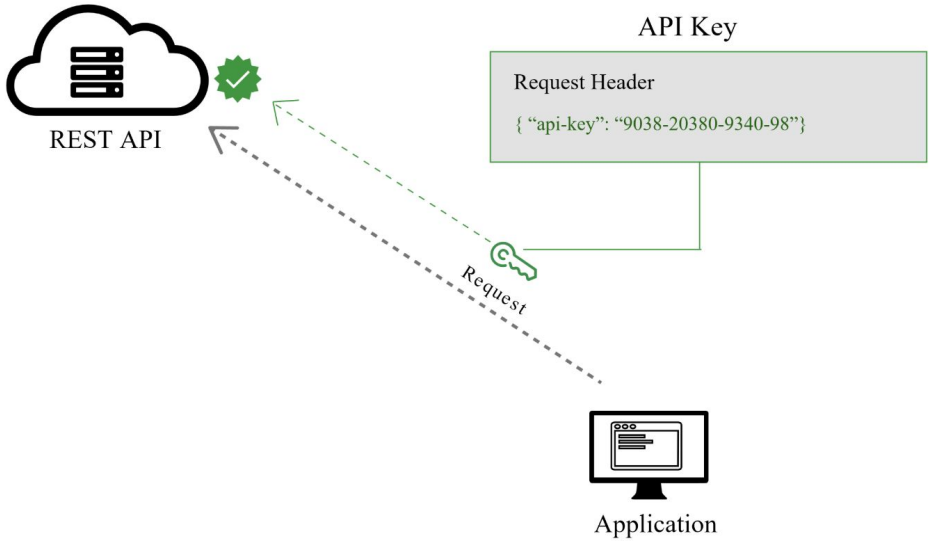
Not Recommended due to security vulnerabilities.

User has **username** and **password**
Token formula: **Base64**(username:password)

In Header:
Authorization: Basic bG6snkjnd4G==

Should only be used HTTPS (SSL)

API Key



Not secure enough

Unique key for each user

If key is stolen, nothing to do

Maybe used in just reads

OAUTH 2.0

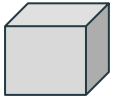
Best Practice for now

Used **JWT (JSON Web Token)** → https://jwt.io/

- It has expire time.
- It is crypted so cant be produced for other users

OAUTH 2.0

User



Server

username
password

give my JWT →

create jwt
with secret key
and expiration

← This is your JWT

/books

Authorization: Bearer JWT →

decode jwt with
secret. validation

← Here all books. You are valid

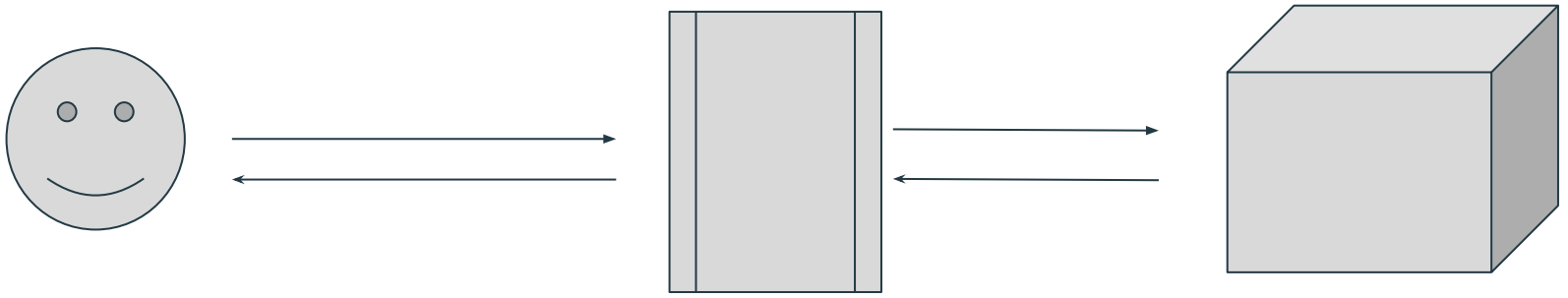
or

← JWT is expired. Grant new one

Middleware

Runs before every request and every response

Manipulate requests or responses in every time



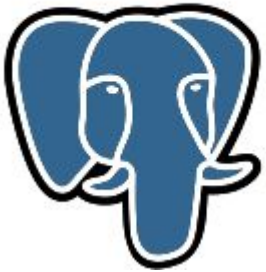
Production Environment

- Create a droplet
- Database and API itself and Load Balancer

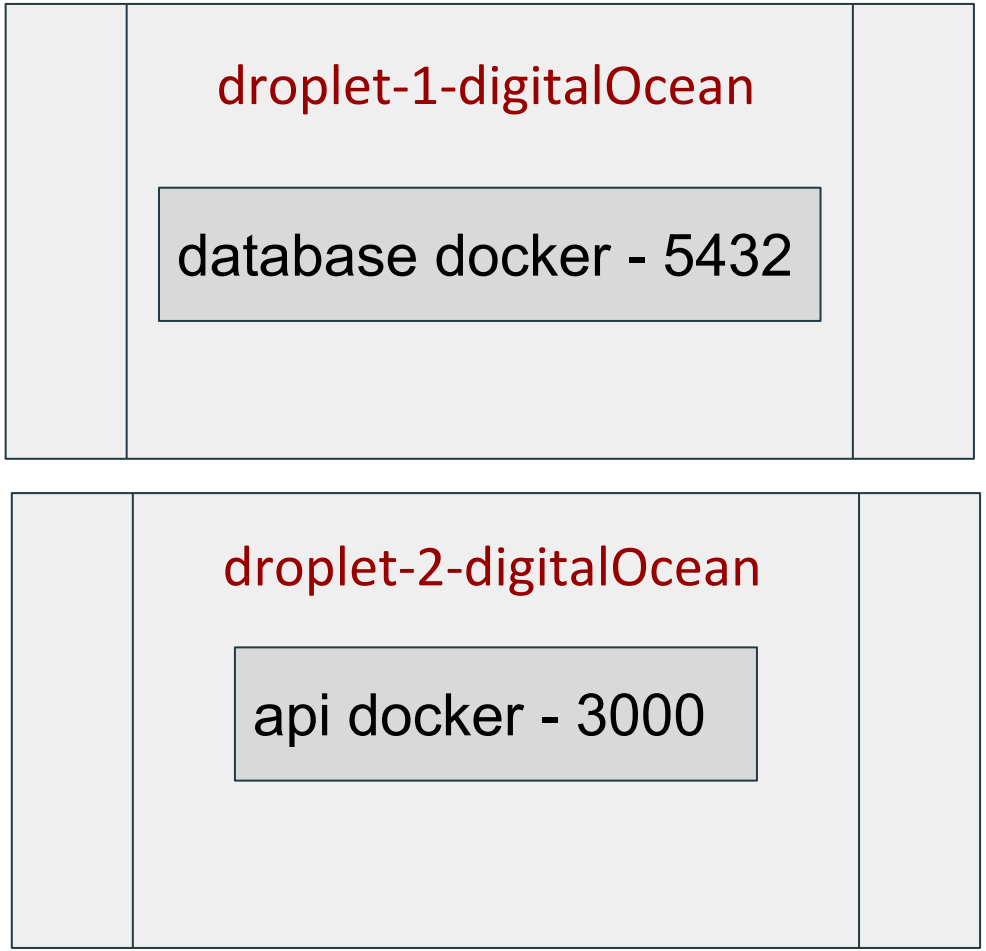
Production Environment

- Easy customizable virtual machines
- Easy to manage
- Dependencies and installments are not trouble

PostgreSQL



- Open source RDBMS Database
- Alternative to MySQL
- Support most of SQL functions

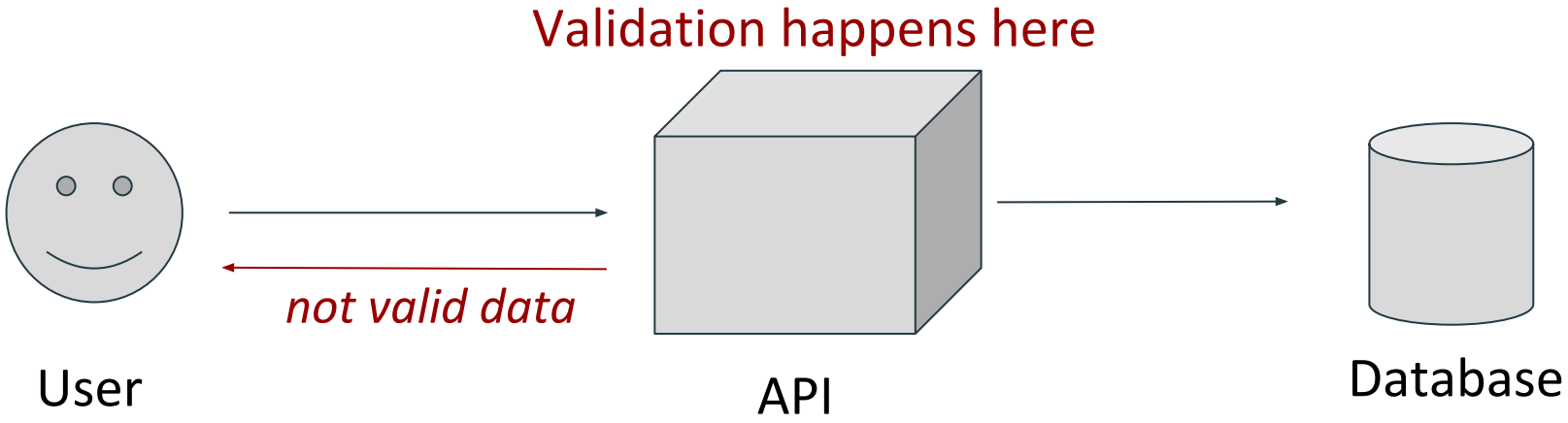

droplet-1-digitalOcean
database docker - 5432
droplet-2-digitalOcean
api docker - 3000

ORM vs Pure SQL

- ORM (Object Relational Model)
- ORM based on objects and methods
- SQL is complex sometime
- Both asynchronous
- SQLAlchemy for ORM

Validation

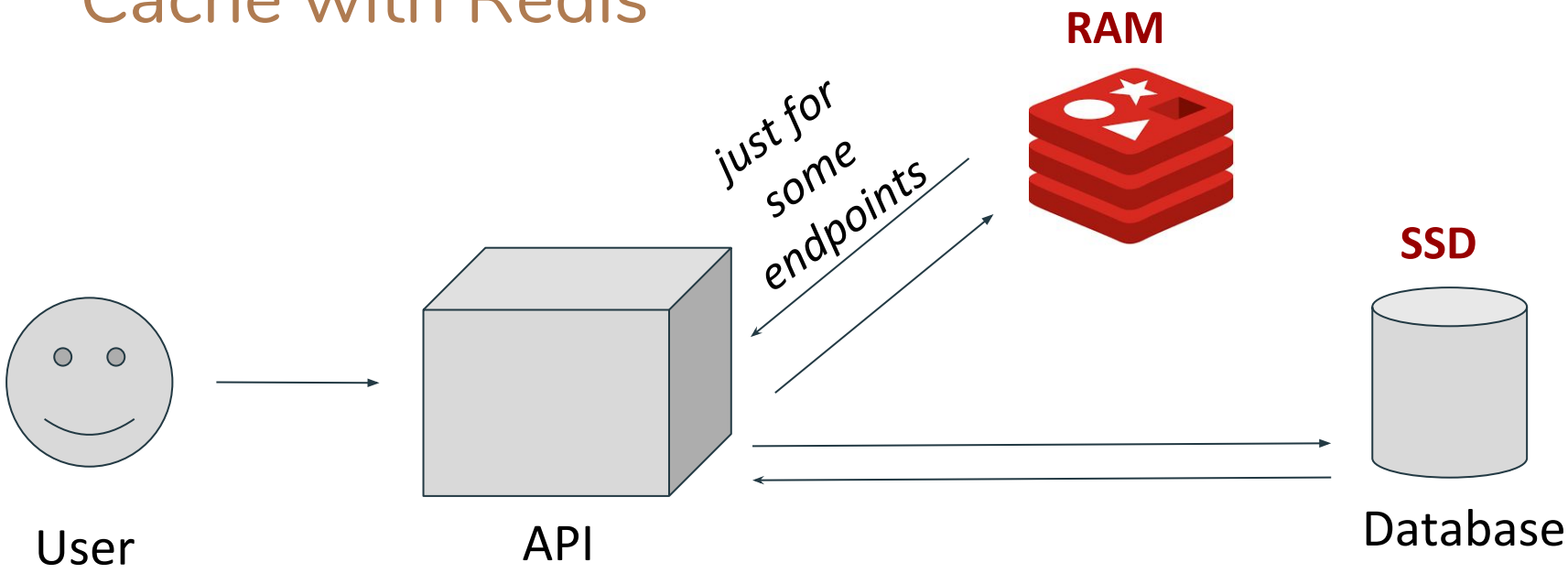
- Validate data types
- Validate with regex
- Validate with enums



Cache with Redis

- Kind of database
- Key-Value storage
- Data is stored in RAM
- Blazingly fast due to RAM
- Keep data with expiration time
- One usage area is in API cache

Cache with Redis



Testing

- Crucial for API development
- Cover all endpoints in testing
- Test database
- Fake values
- Test both positive and negative scenarios

Load Testing

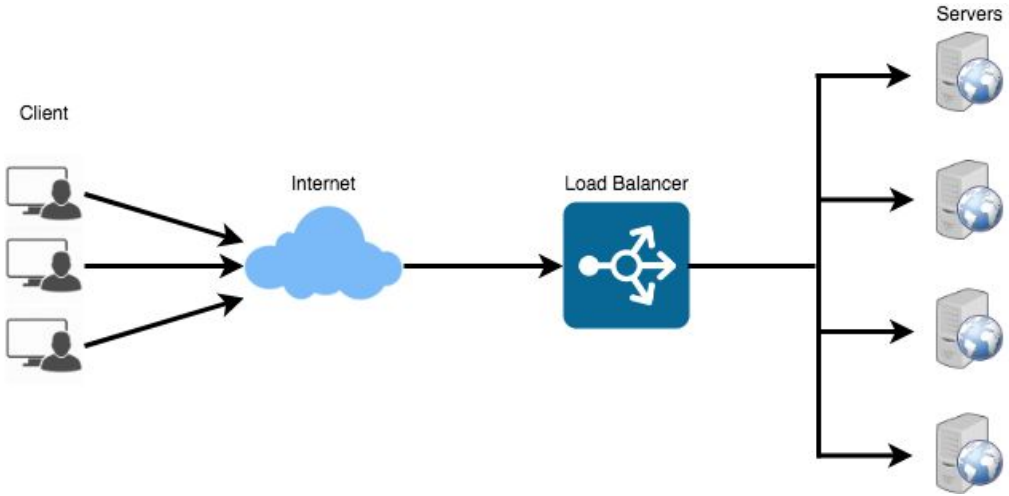
- How much request you API can handle in a second?
- Response time of the endpoints
- Average response time
- Maximum limit

Deploying

- We deploy the api to server
- Docker is feasible way for deploying
- API server will run inside docker
- You can make your deploying process “semi-automate“

Load Balancer

- Between user and API server
- Distribute the traffic
- Decreases the wait time
- Request per second increases

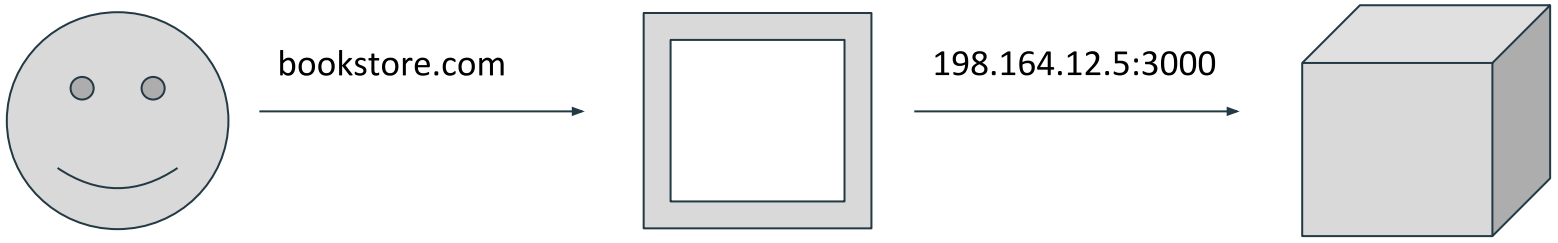


Reverse Proxy

- Middleware between user and backend system
- Security, Reliability, Performance, Rule based request
- HTTPS-SSL layer
- IP Restriction, Rate Limit, Port restriction etc.
- Load Balancer Layer
- Redirection Layer
- Auto caching layer, especially for frontend

NGINX

Reverse Proxy



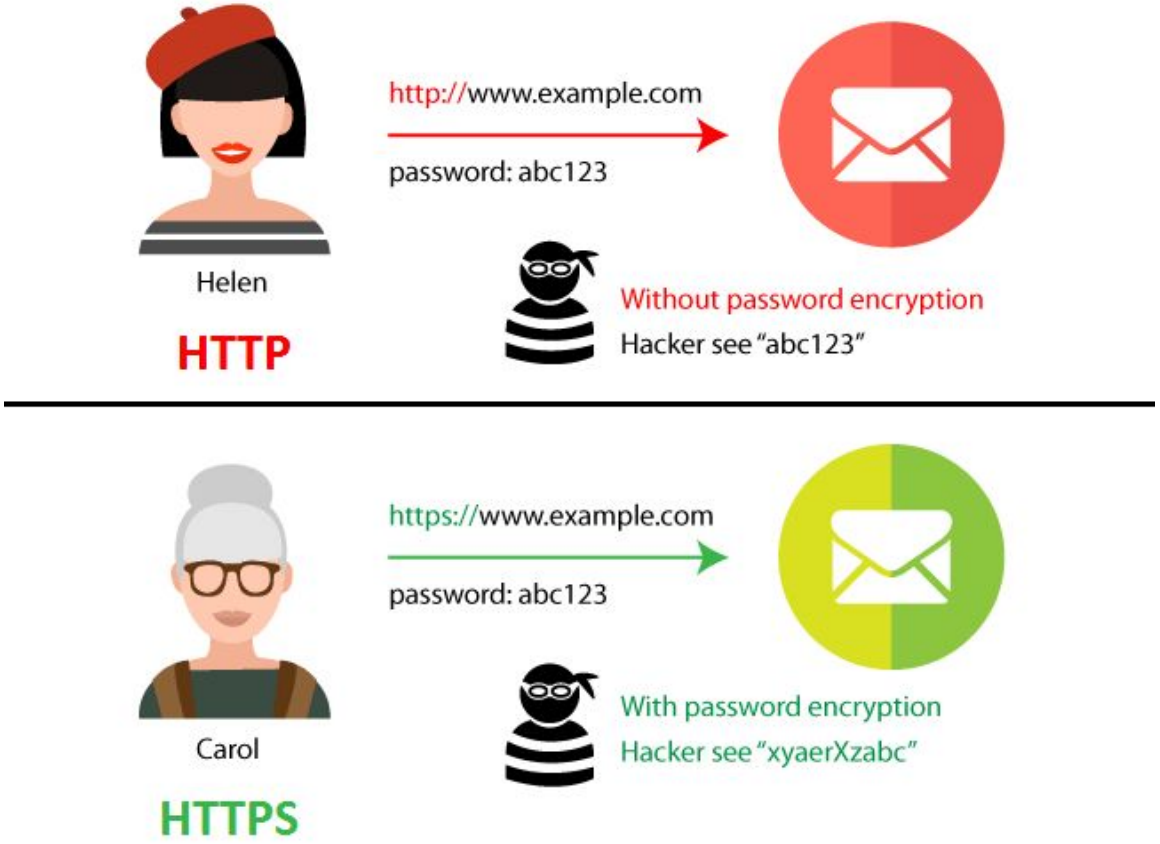
HTTPS-SSL
IP Restriction
Rate Limit
Auto Cache
Load Balance

HTTPS

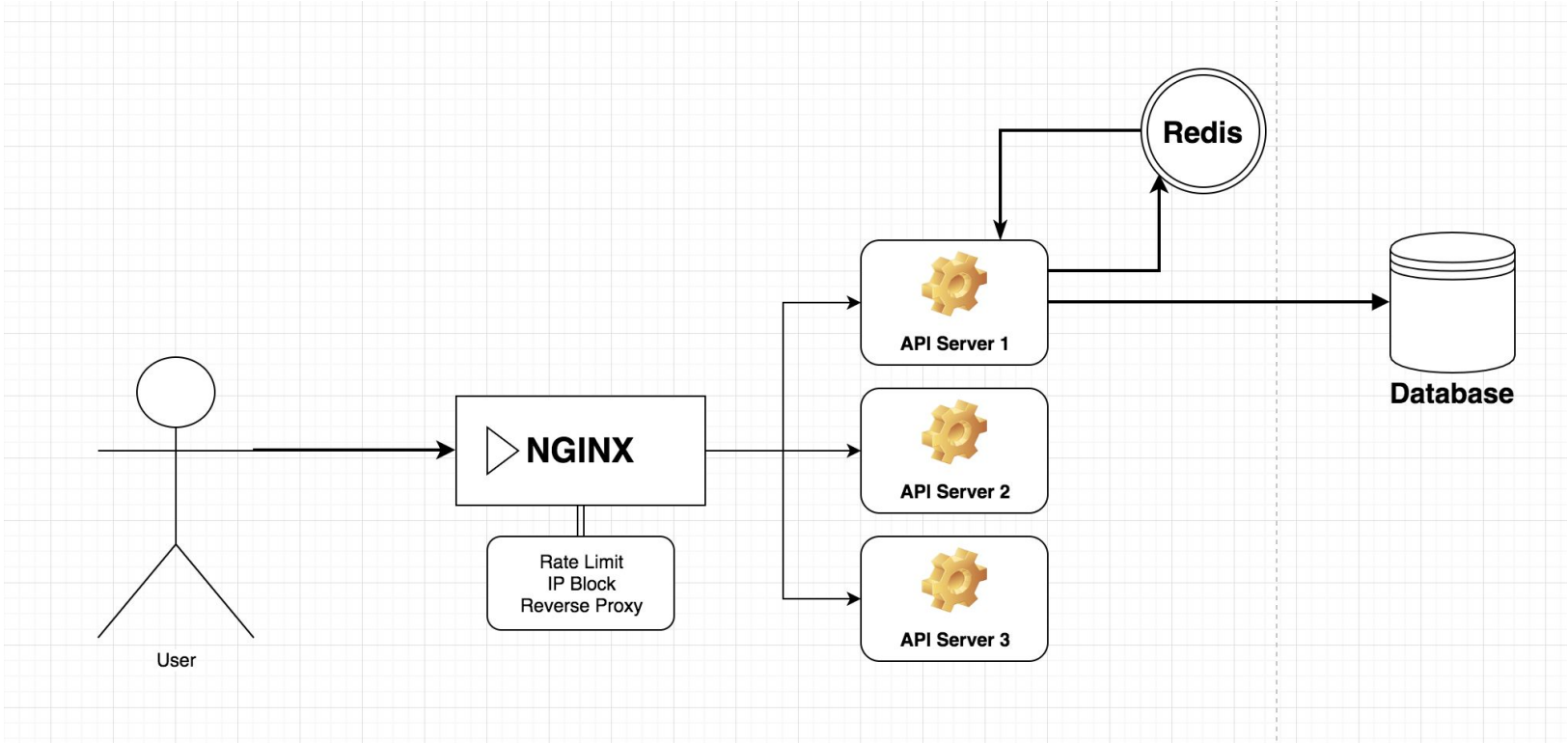
- Encrypted requests
- Apikeys, JWT Tokens, Passwords are in safe
- Security layer
- API must have HTTPS

Free SSL Tool **or** You can buy from domain service providers

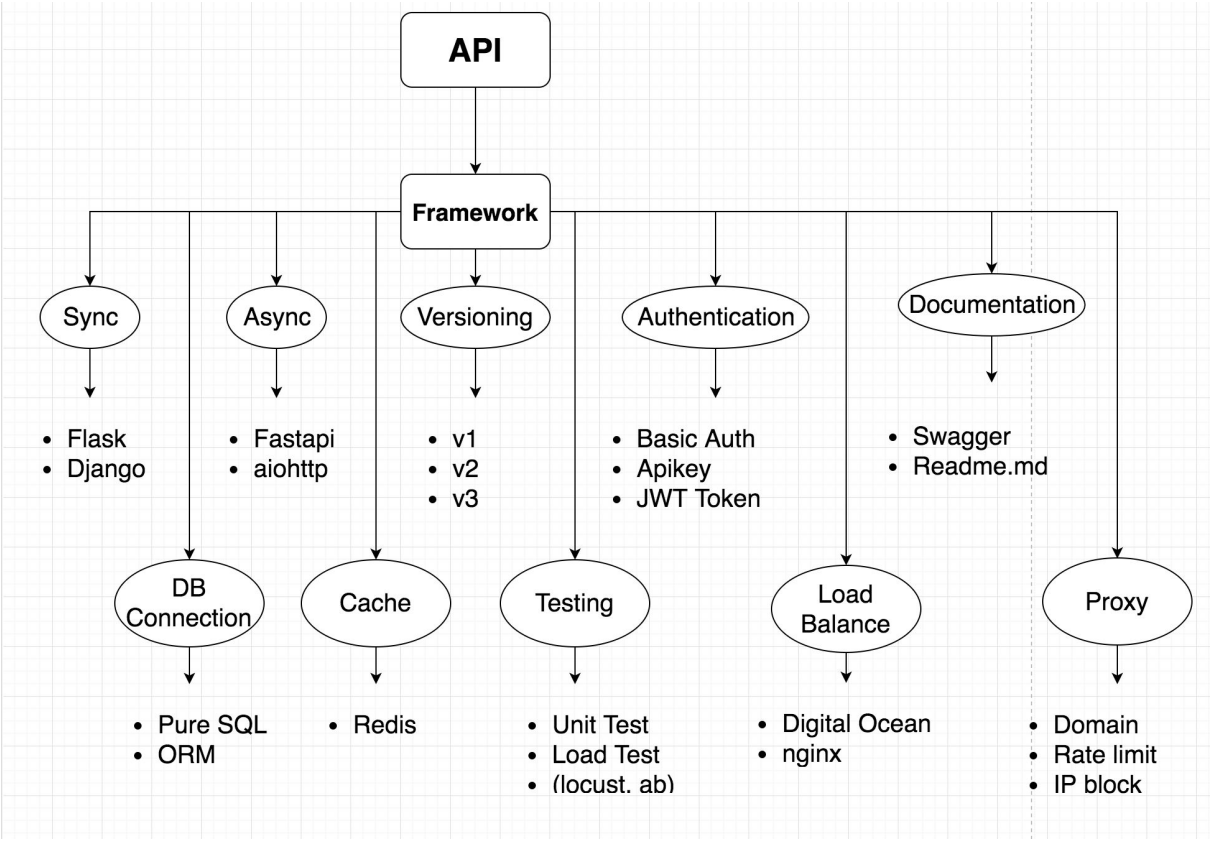
HTTPS


http://www.example.com
password: abc123
Helen
HTTP
Without password encryption
Hacker see "abc123"
https://www.example.com
password: abc123
Carol
HTTPS
With password encryption
Hacker see "xyaerXzabc"

Backend Architecture



Backend Architecture



References

- Slide 4: http://ddi-dev.com/uploads/media/news/0001/01/f2da1c598e2ff9bc29b229773a189d33d38e0252.jpeg
- Slide 7:https://medium.com/@perrysetgo/what-exactly-is-an-api-69f36968a41f
- slide10: https://miro.medium.com/max/1080/1*t_oCyHBstMnF8WpZ67pKTg.jpeg
- slide11:https://www.techempower.com
- slide18: https://codeteddy.com/2017/06/06/create-api-with-asp-net-core-day-3-working-with-http-status-codes-in-asp-net-core-api/
- slide 28: https://www.akana.com/solutions/api-security
- slide 31,33: https://blog.restcase.com/4-most-used-rest-api-authentication-methods/
- slide 37: https://www.iconfinder.com/icons/389016/accept_approve_available_check_complete_confirm_correct_environmental_green_healthy_mark_ok_okay_tick_yes_icon
- slide 45: https://kubedex.com/resource/locust/ , http://devnot.com/etiket/apache-bench/
- slide 47: https://www.educative.io/courses/grokking-the-system-design-interview/3jEwl04BL7Q
- slide 51:https://seopressor.com/blog/http-vs-https/