

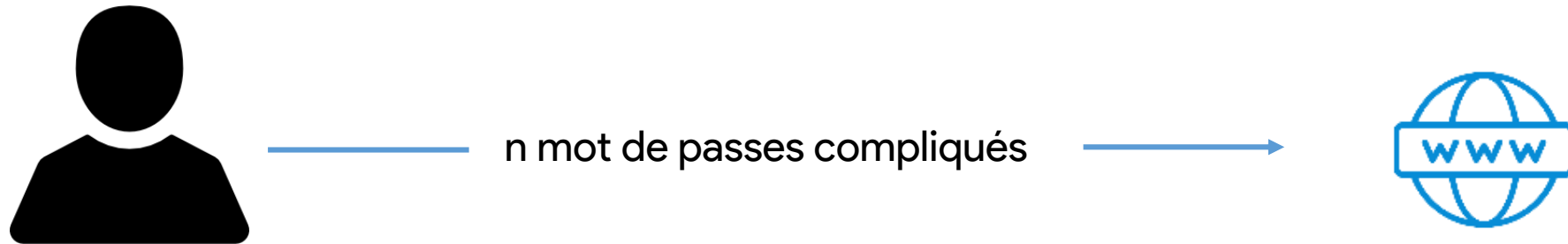


Face Key

Tuteur : Pierre Andry

Quentin Gerard & Louis L'Haridon & Matthieu Vilain

Animation




Animation



Animation



Site internet 

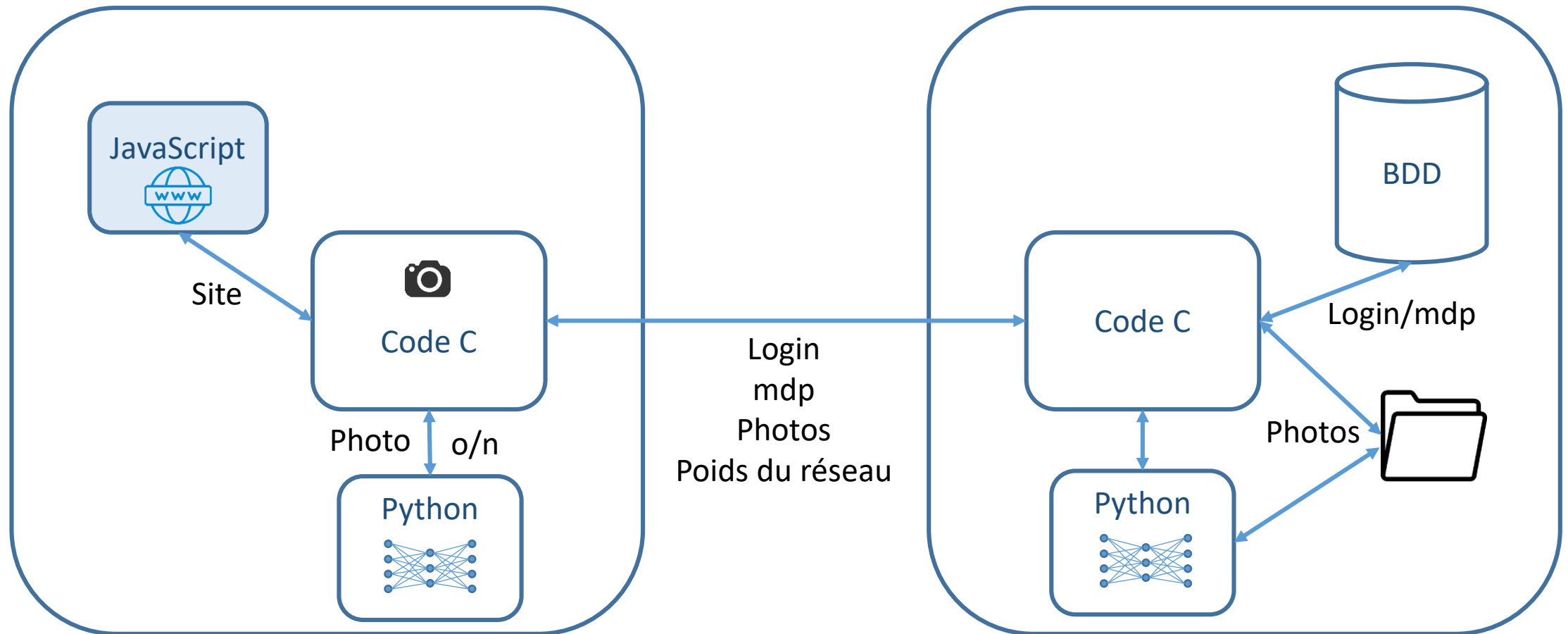


Site internet

Architecture

Client

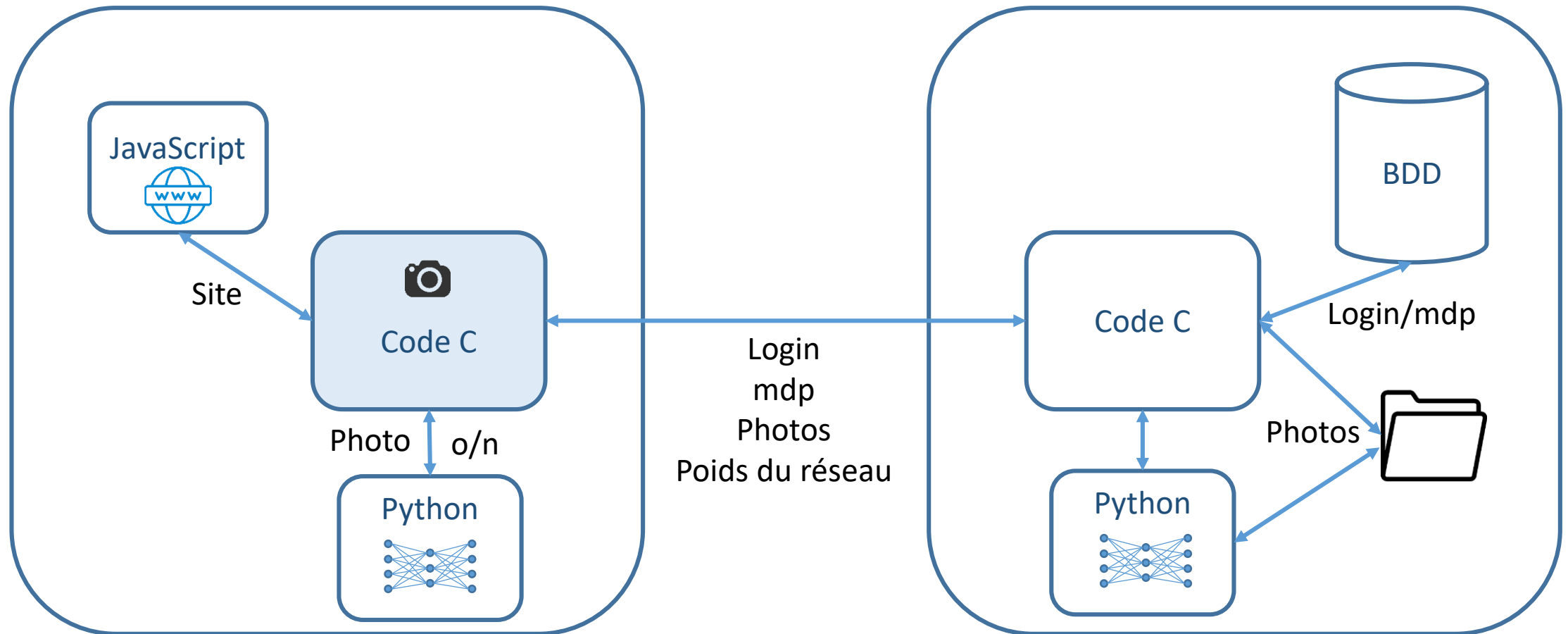
Serveur



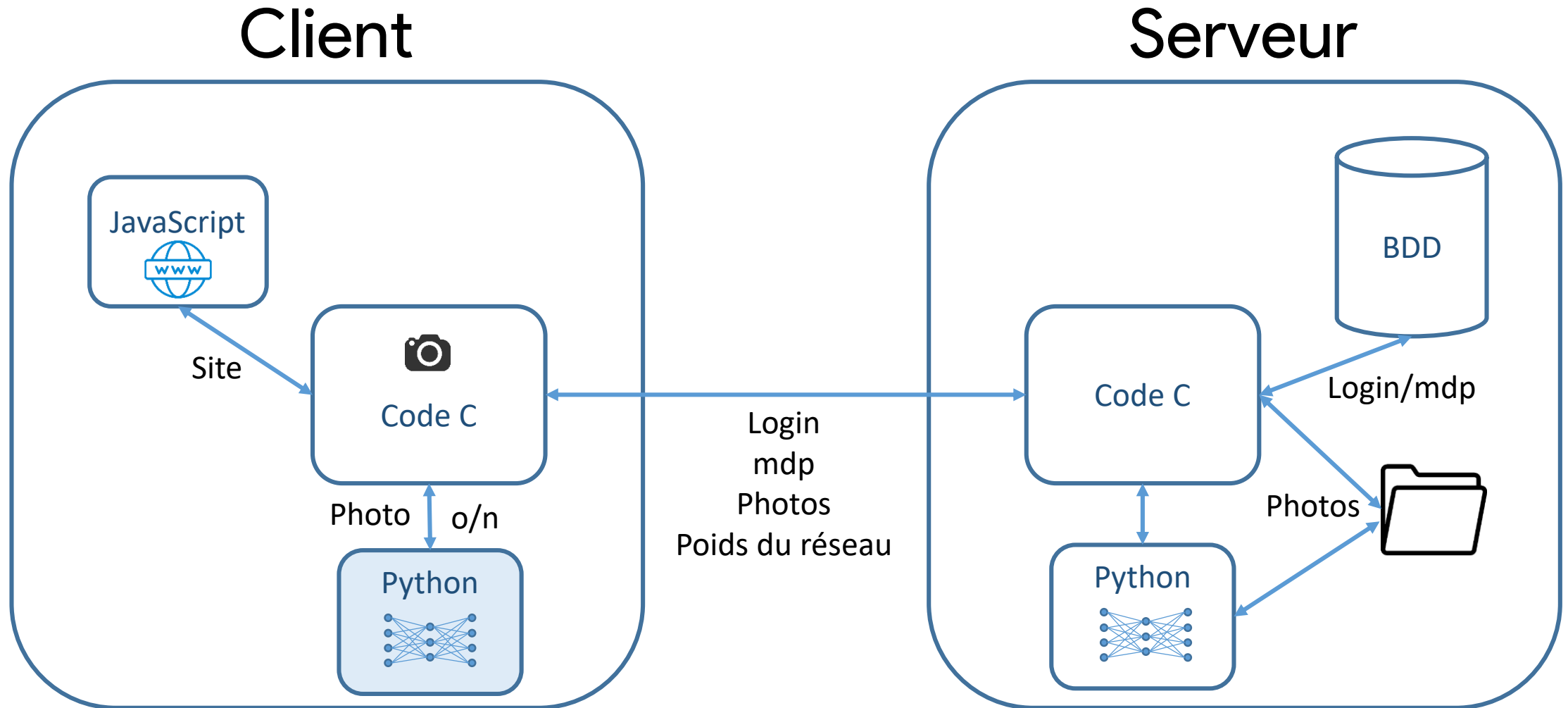
Architecture

Client

Serveur



Architecture



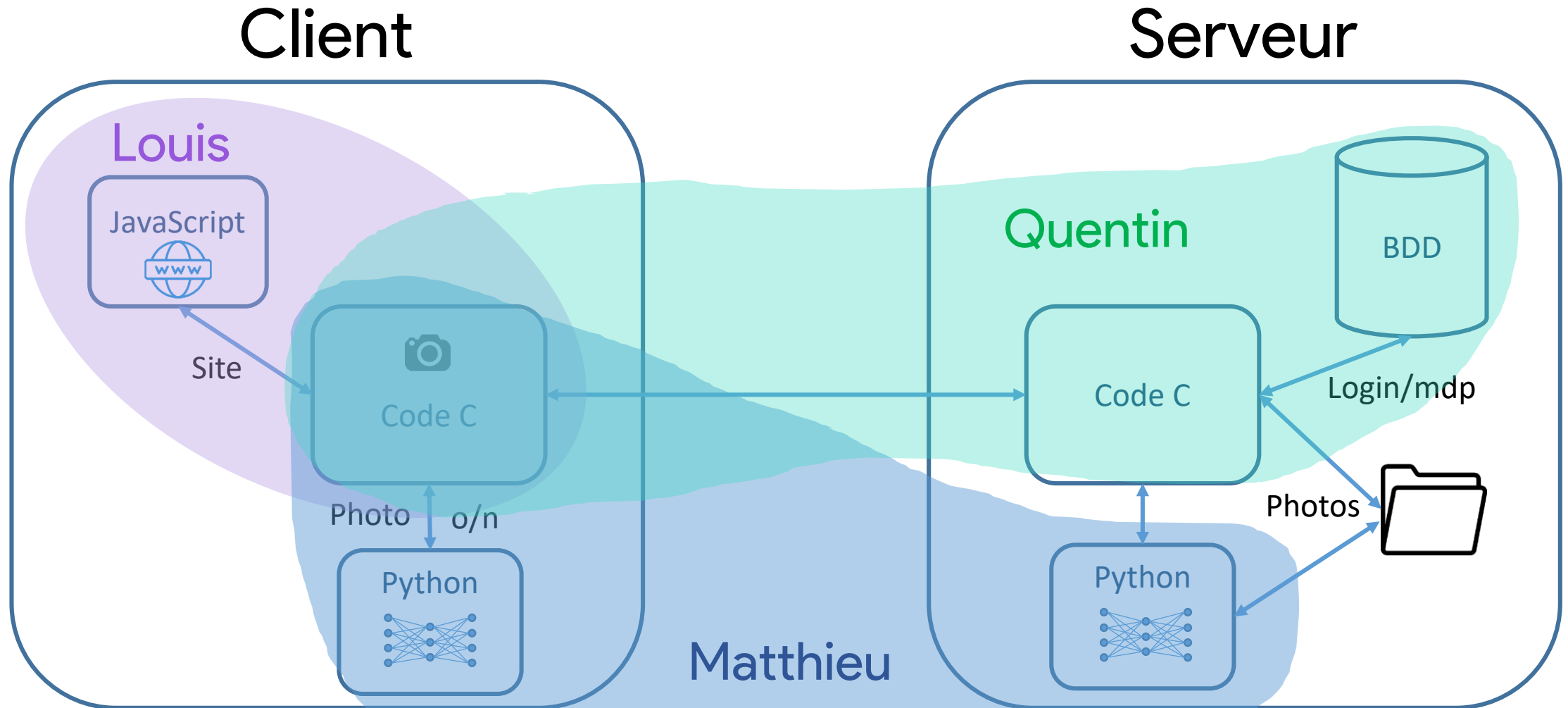
GPI : Répartition des tâches

Louis : Interface Utilisateur

Matthieu : Reconnaissance Faciale

Quentin : Sécurité/cryptographie

GPI : Répartition des tâches





Interface utilisateur

Objectif : rendre la connexion aux sites très simple

Interface

Comment faciliter la connexion utilisateur?

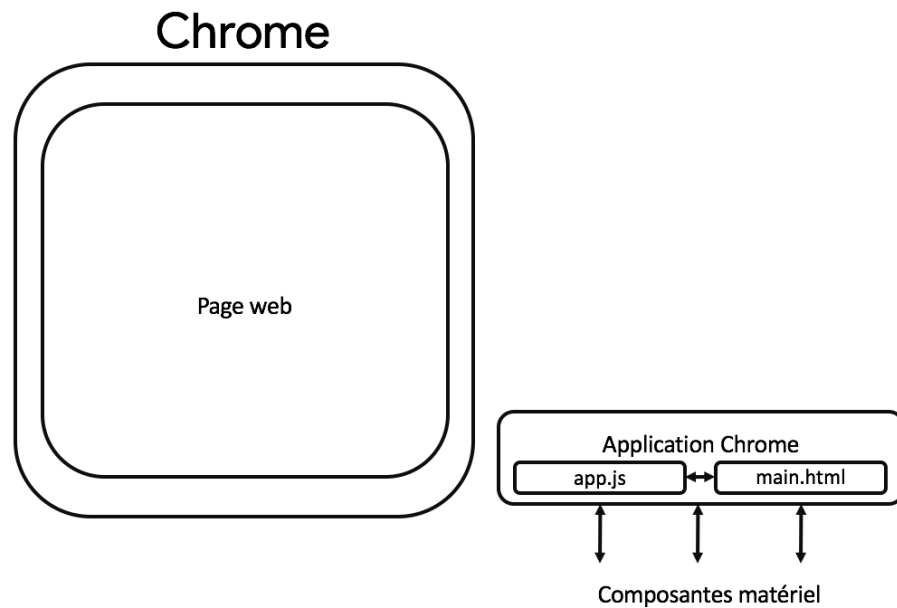
Choix techniques

Source	Chrome	Safari	Firefox	IE + Edge	Opera	UC Browser	Android	Autres
W3Counter	59,9 %	15,7 %	8,5 %	7,3 %	3,4 %	1,5 %	0,0 %	3,7 %

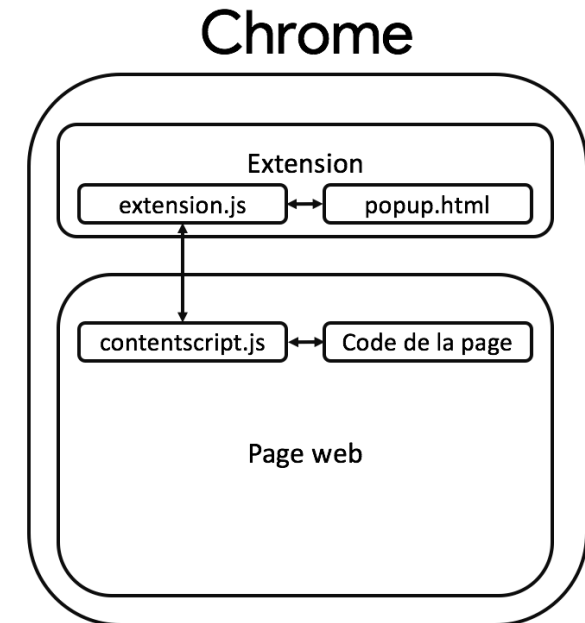
Technologie	Chrome extension	Safari extension
Récupération du contenu d'une page	✓	✓
Injection de contenu dans la page	✓	✓
UDP	✓	✗

Chrome : technologies

Application Chrome

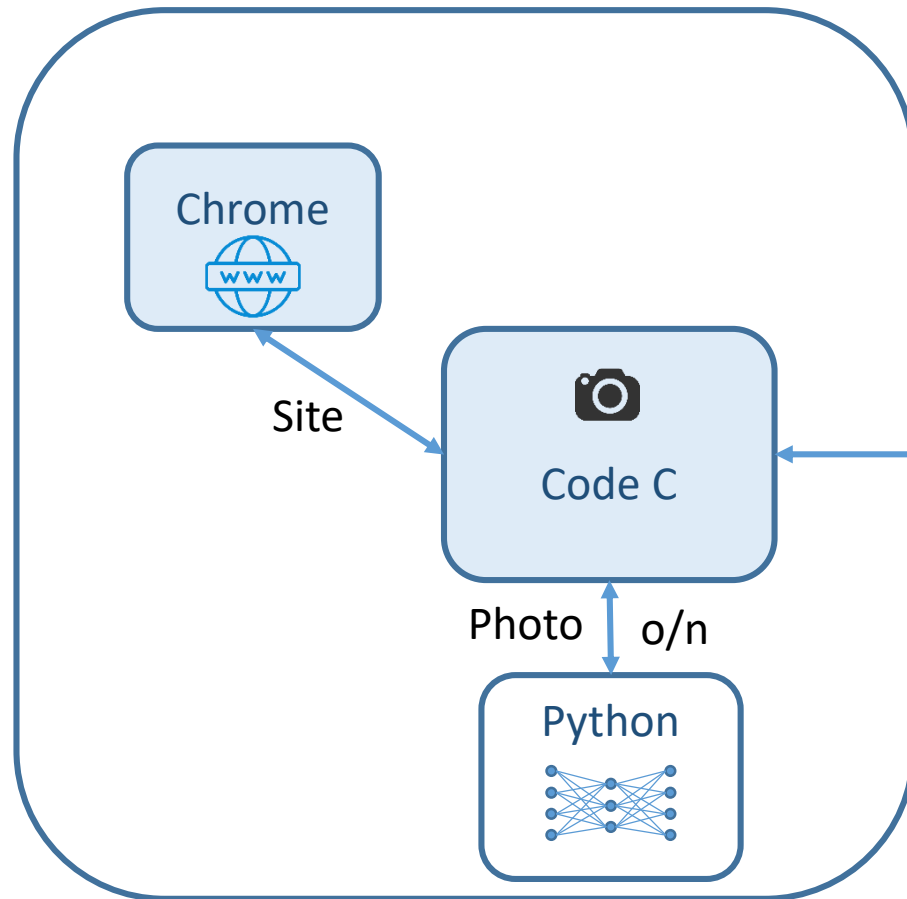


Extension Chrome

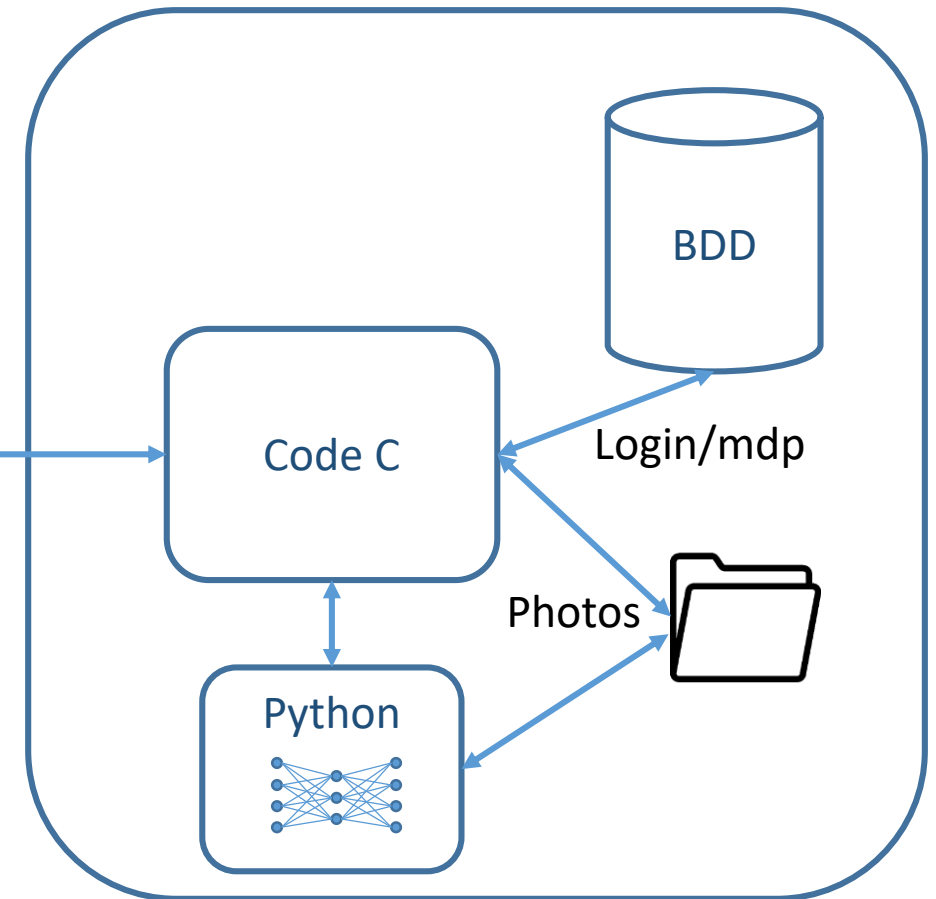


Intégration de l'IHM

Client



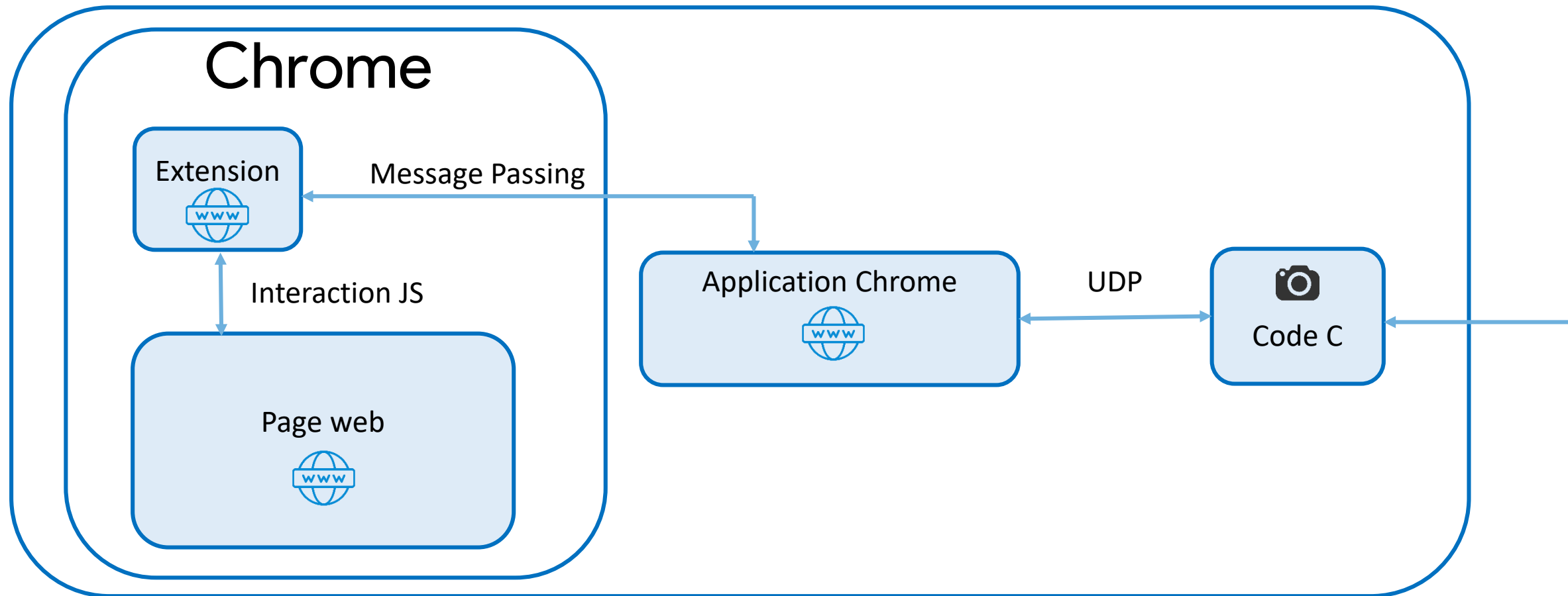
Serveur



Login
mdp
Photos
Poids du réseau

IHM : Connexion via le client

Client



Bilan avancement

Ce qui est fonctionnel

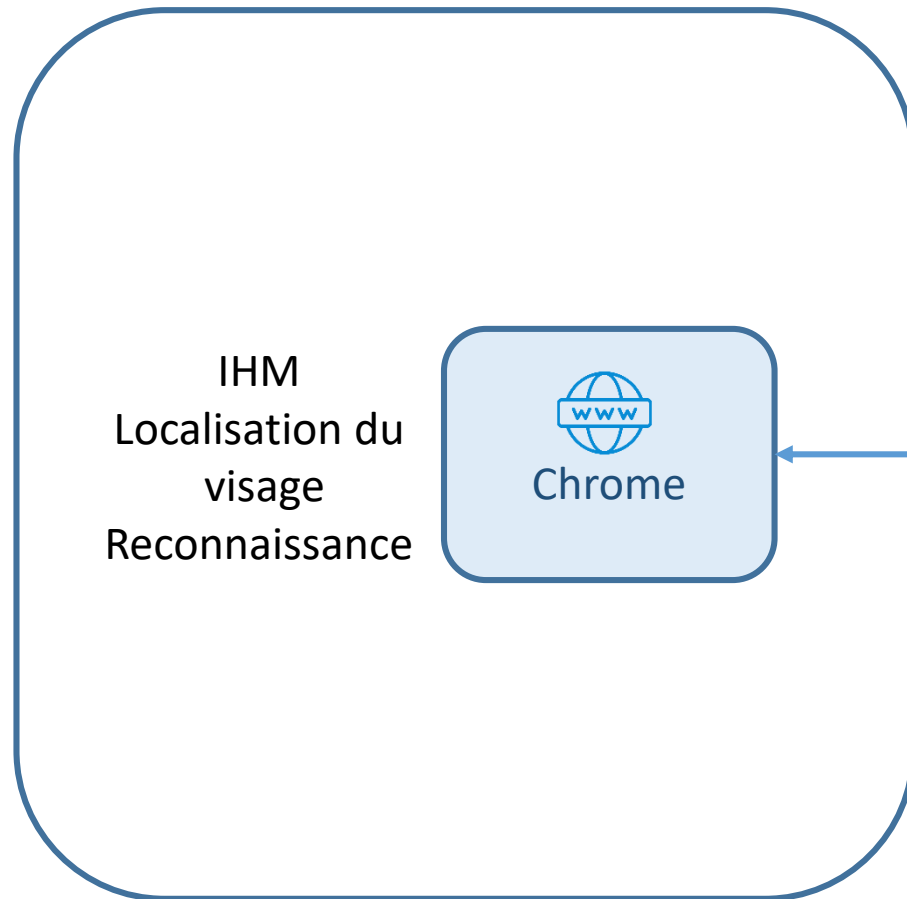
- ✓ Interaction avec la page web
- ✓ Communication extension/application
- ✓ UDP vers C
- ✓ UDP depuis C
- ✓ Connexion simple

Ce qu'il reste à implémenter

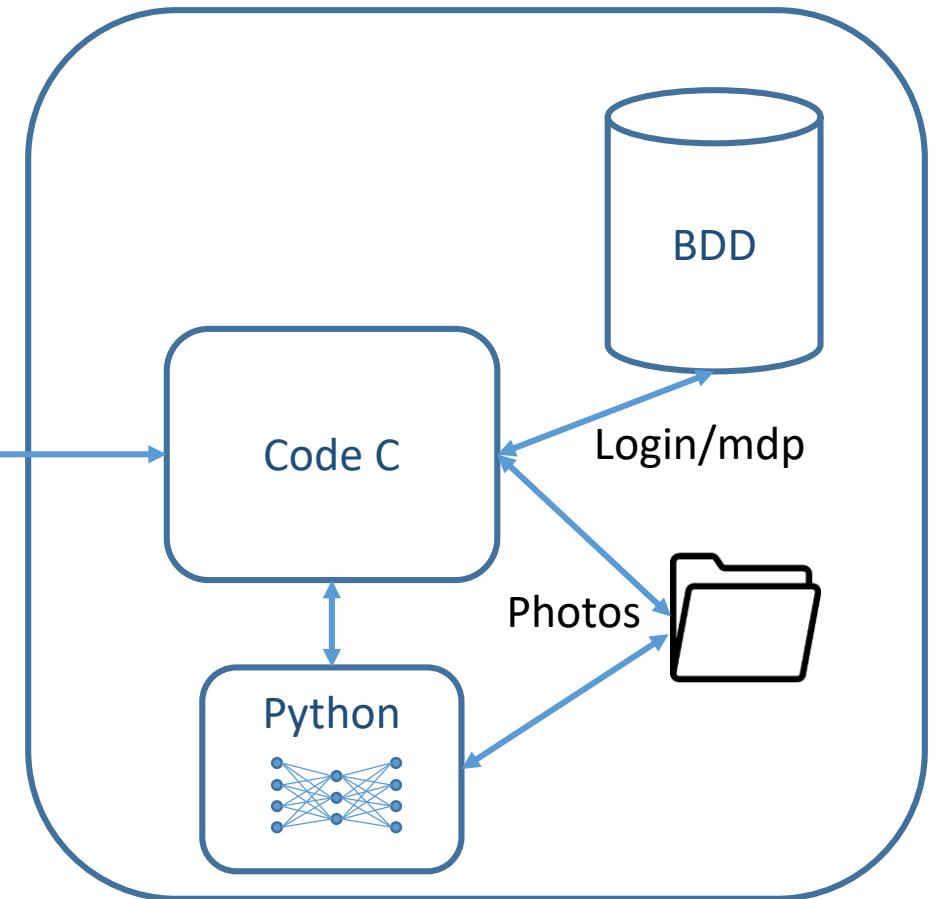
- ✗ Choix du compte
- ✗ Connexion à Face Key
- ✗ Gestion détaillée des erreurs

Perspectives

Client



Serveur



Login
mdp
Photos
Poids du réseau

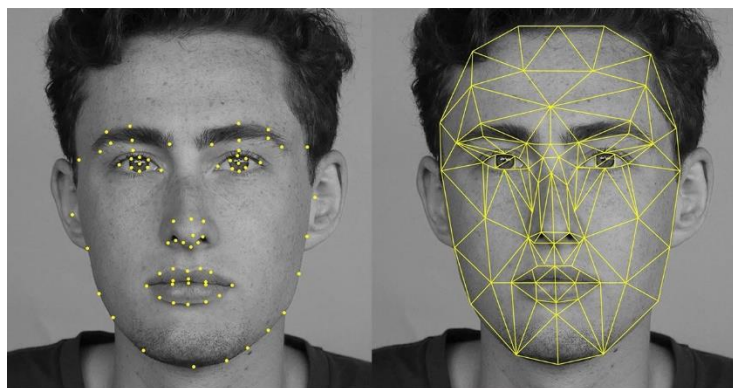


Reconnaissance faciale

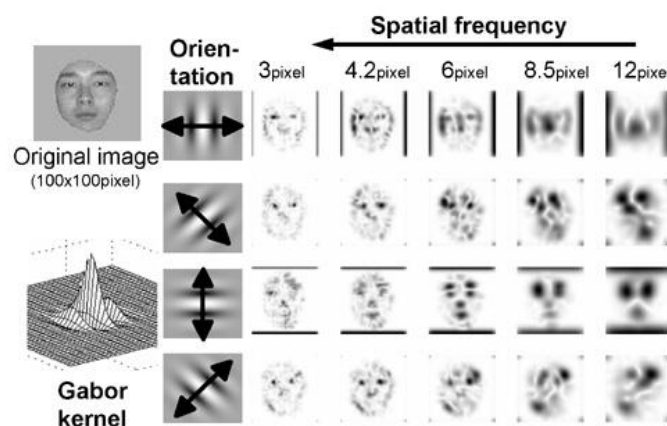
Objectif : rendre la connexion aux sites fiable et sécurisé

Bibliographie

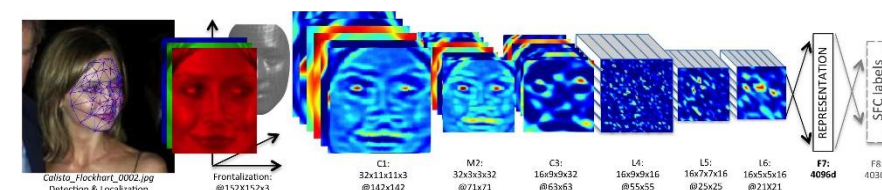
Landmark détection 1971



Feature détection 1993



Deeplearning 97,35% 2014 FaceNet 99,63% 2015

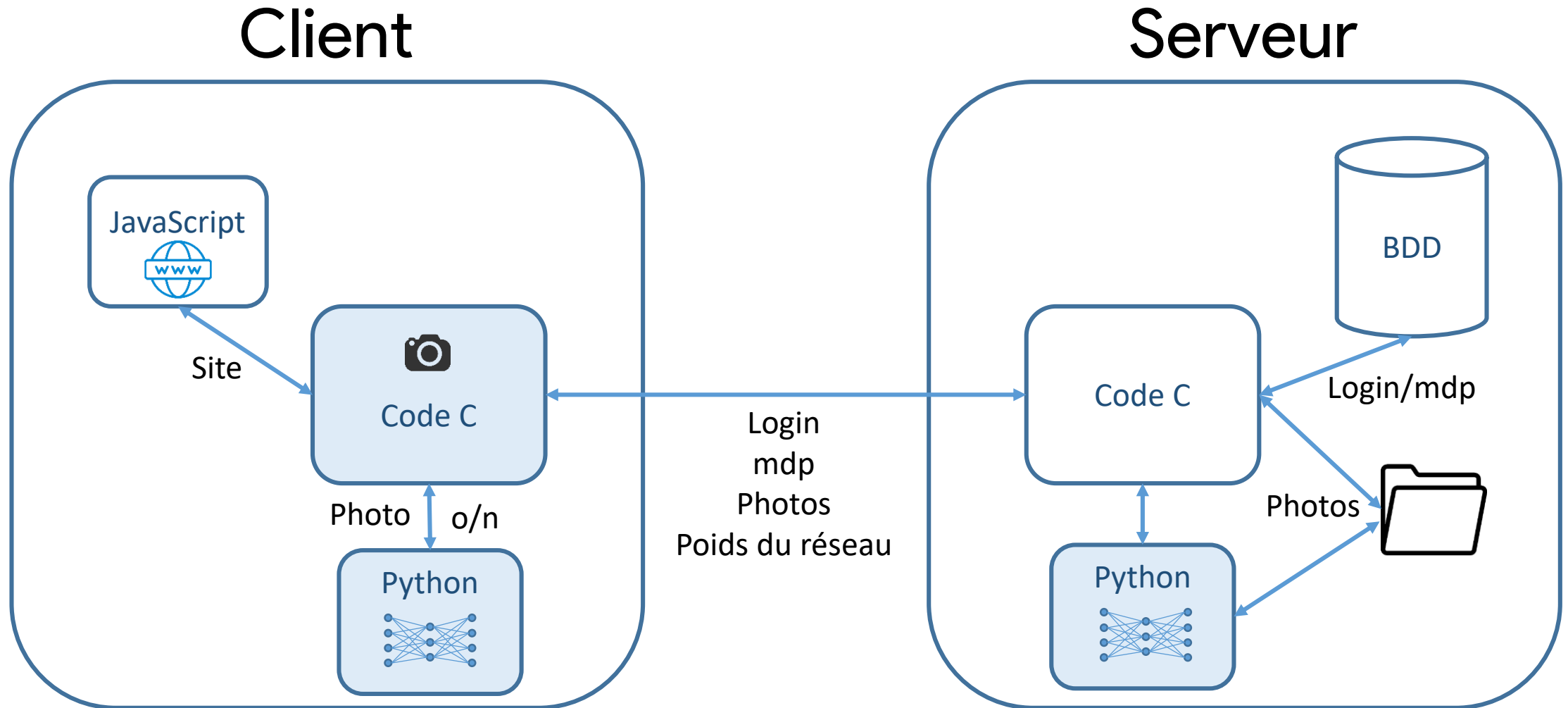


J. Goldstein, L. D. Harmon, and A. B. Lesk, "Identification of human faces" 1971.

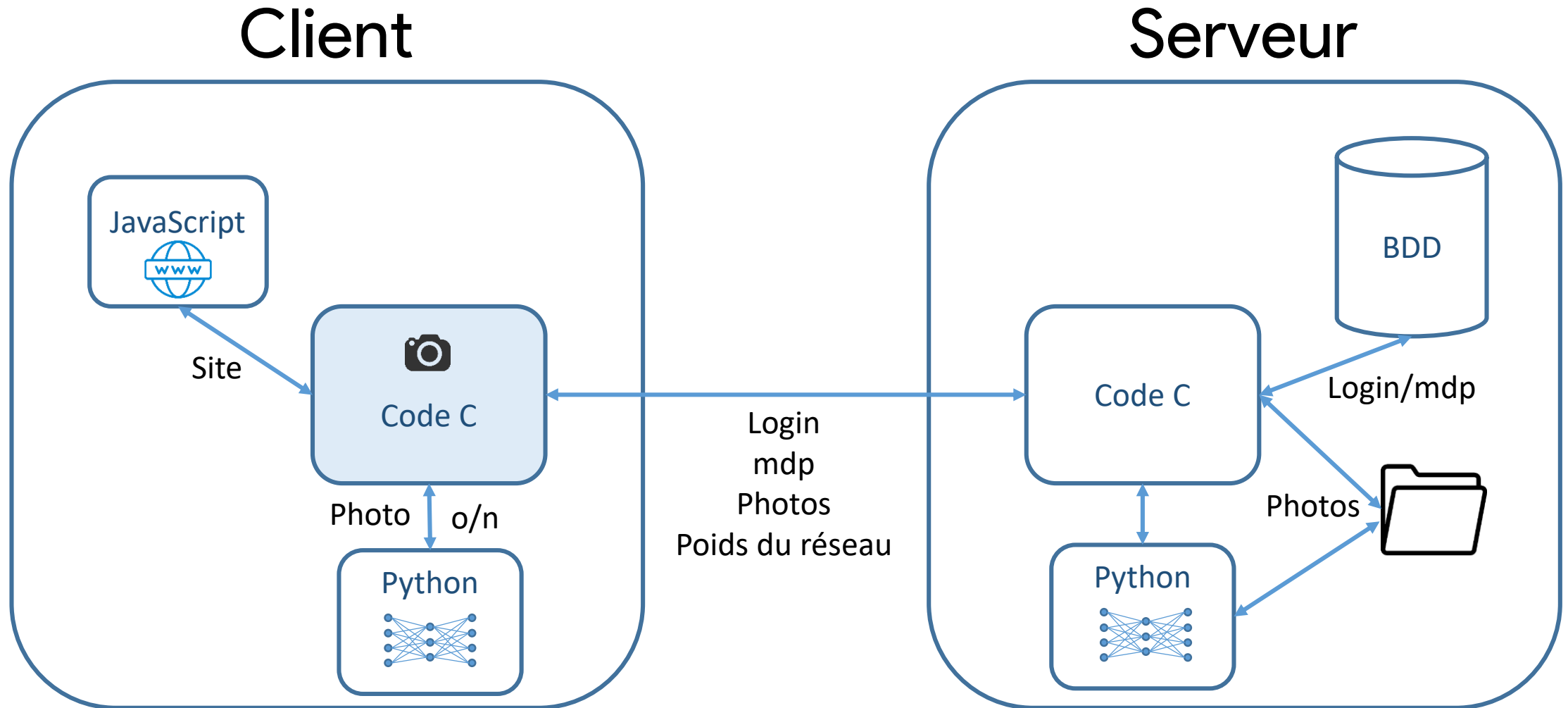
Y. Taigman, M. Yang, M. Ranzato and L. Wolf " DeepFace: Closing the Gap to Human-Level Performance in Face Verification " 2014

R. Brunelli and T. Poggio, "Face Recognition: Features versus Templates" 1993.

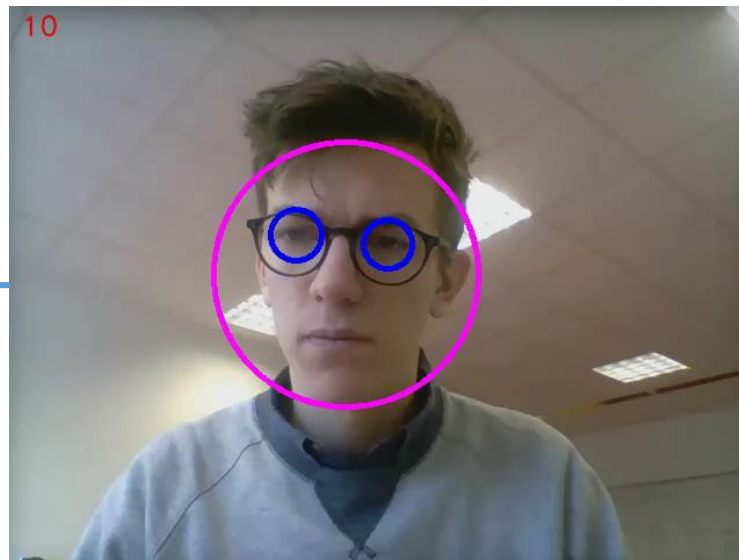
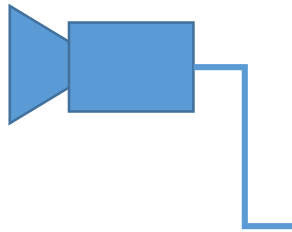
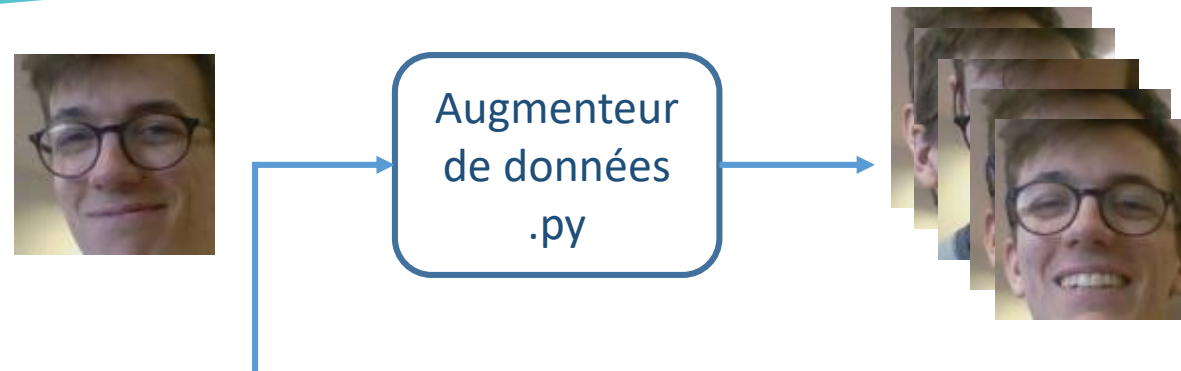
Intégration de la reconnaissance



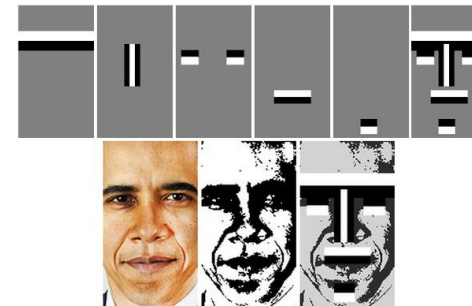
Intégration de la collecte de données



Protocole de collecte



Haar cascade

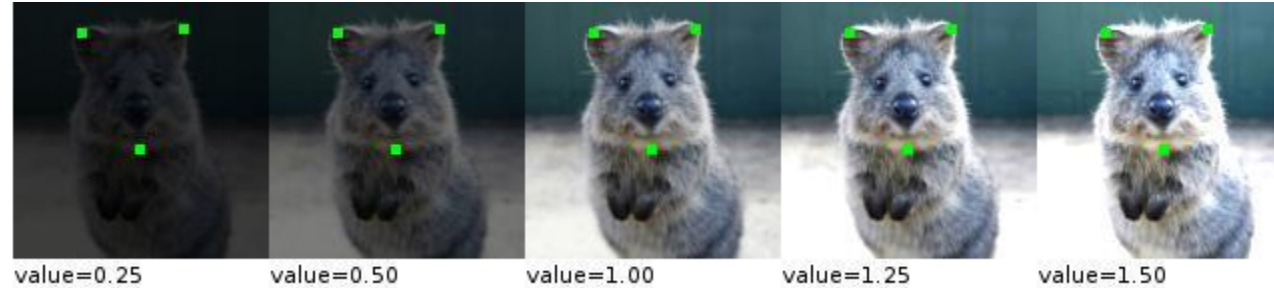


251;15;200;103;70
50;1;23;125;210;1
63;135;16;32;156;
120;231;152;15;10
2;256;24;123;20;1
173;125;236;41;15
42;123;21;45;214;
61;124;120;23;89
38;213;21;42;98;7

zero mean
Equal variance
Shuffle

Augmentation d'image

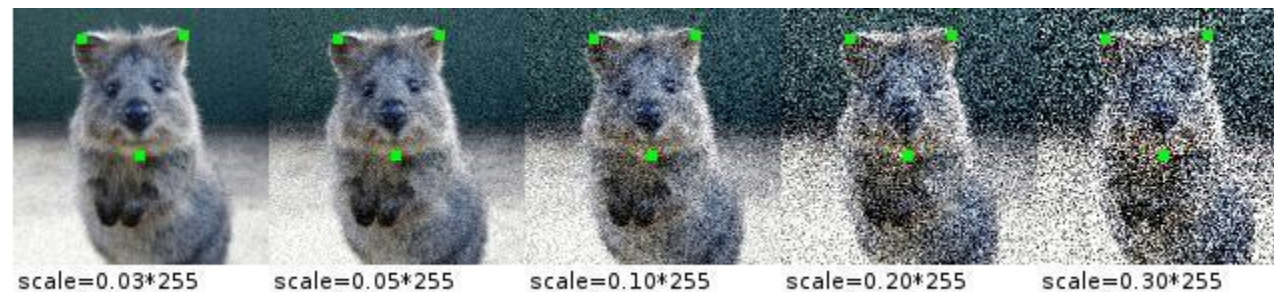
Multiply



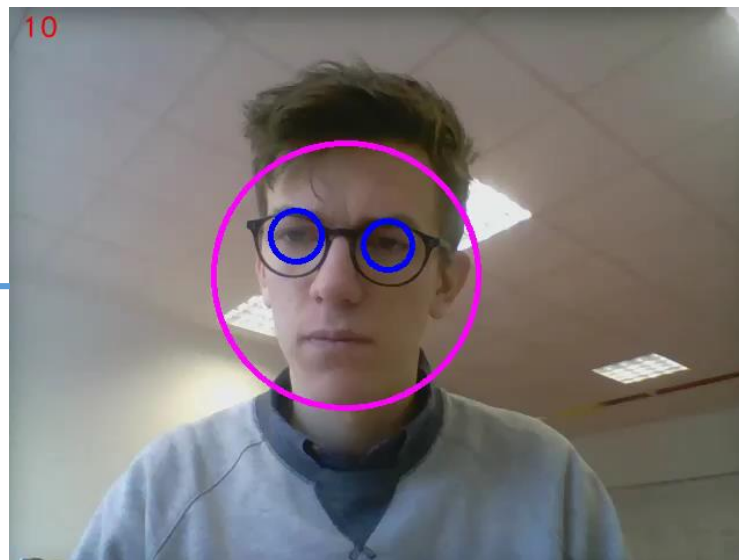
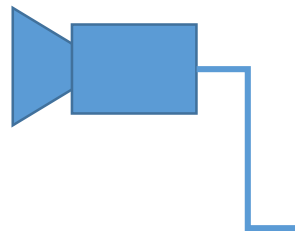
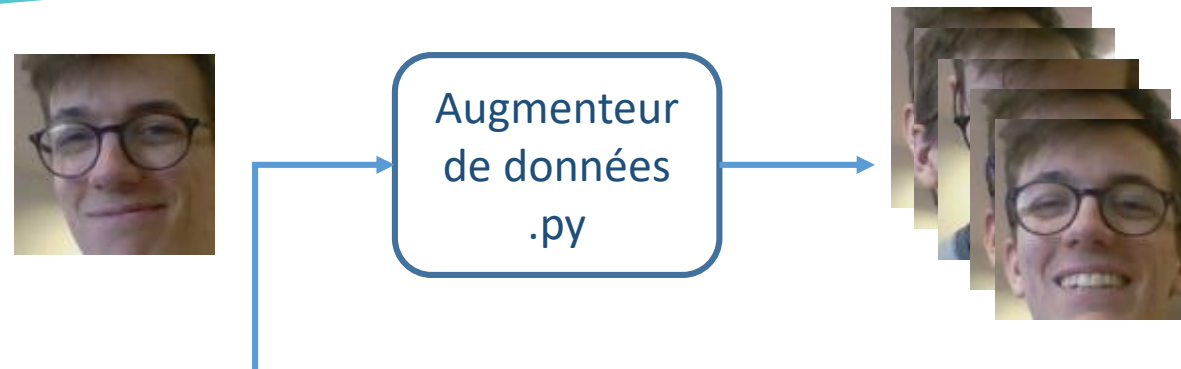
GaussianBlur



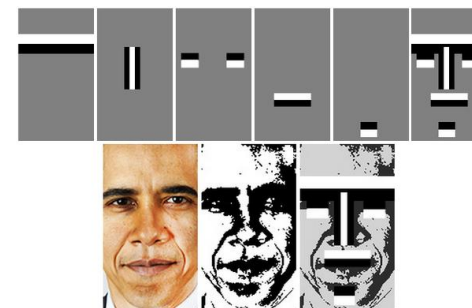
AdditiveGaussianNoise



Protocole de collecte



Haar cascade



251;15;200;103;70
50;1;23;125;210;1
63;135;16;32;156;
120;231;152;15;10
2;256;24;123;20;1
173;125;236;41;15
42;123;21;45;214;
61;124;120;23;89
38;213;21;42;98;7

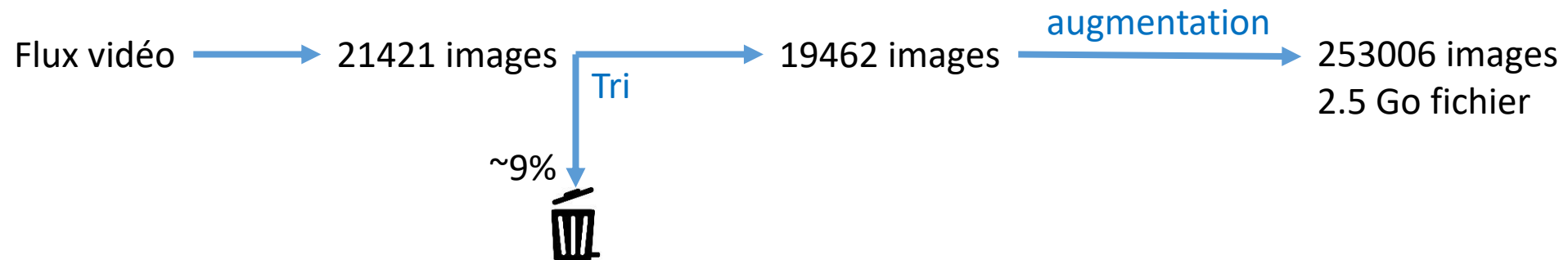
zero mean
Equal variance
Shuffle

Détails base de données

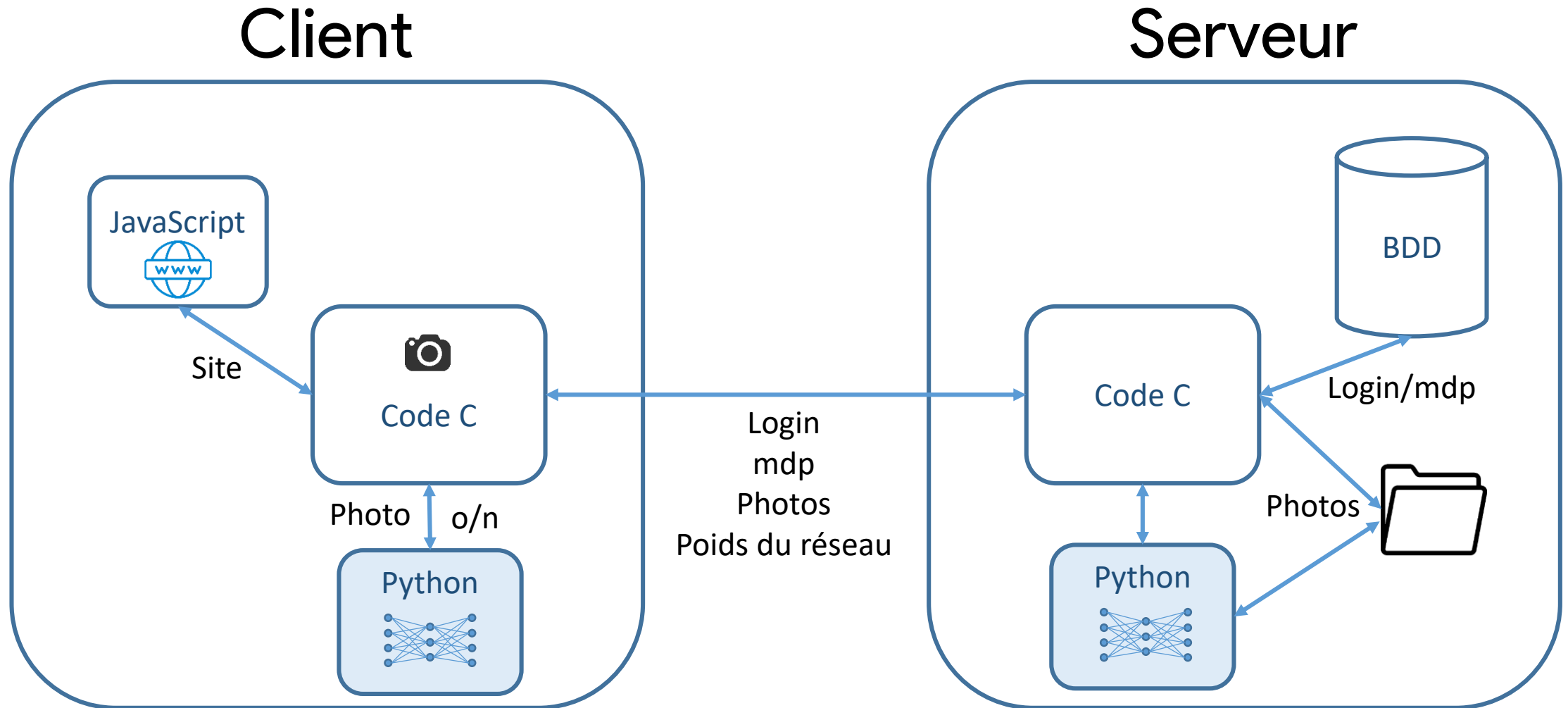
17 classes



100x100pixels

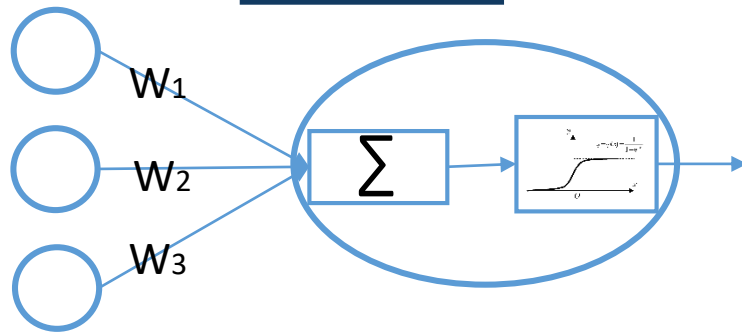


Intégration de la reconnaissance



Solution 1 : Perceptron multicouches

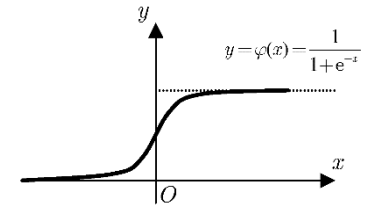
Neurone



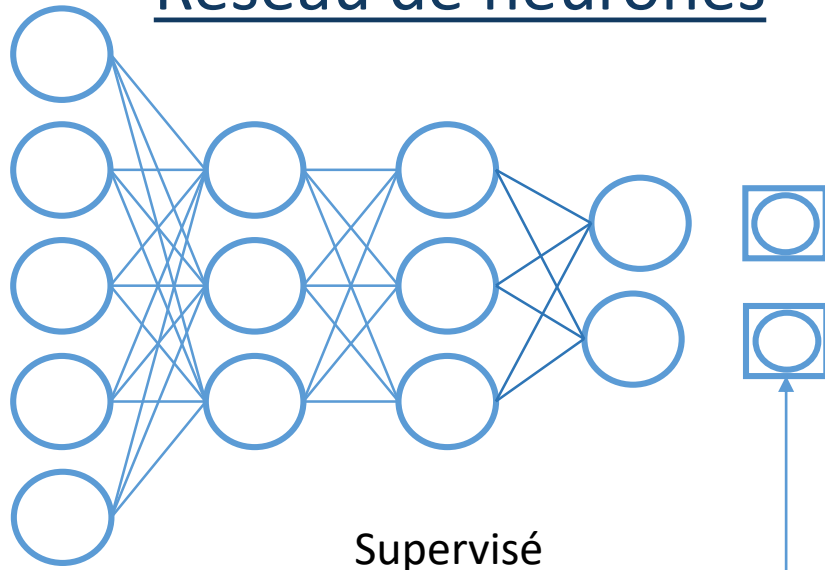
Feed forward

$$Pot_j = \sum_i w_{ij} \times x_i$$

$$\text{Signalout} = \text{sigmoid}(Pot)$$



Réseau de neurones



Back propagation

$$E_i^{out} = \text{superviseur}_i^d - \text{sig}_i^{out}$$

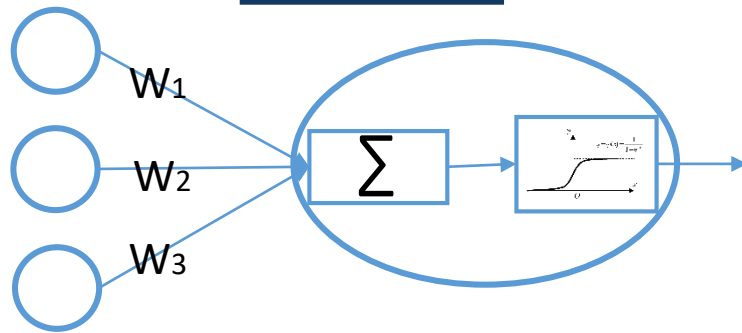
$$E_i^n = \sum_j w_{ij} \times \delta_j^{n+1}$$

$$w_{ij}^{t+1} = w_{ij}^t + \eta \times \text{sig}_i^n \times \delta_j^{n+1}$$

$$\delta^n = \text{sig}^n \odot (1 - \text{sig}^n) \odot E^n$$

Solution 1 : Perceptron multicouches

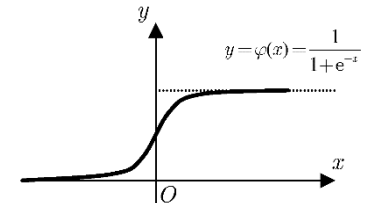
Neurone



Feed forward

$$Pot_j = \sum_i w_{ij} \times x_i$$

$$Signalout = \text{sigmoid}(Pot)$$



Back propagation

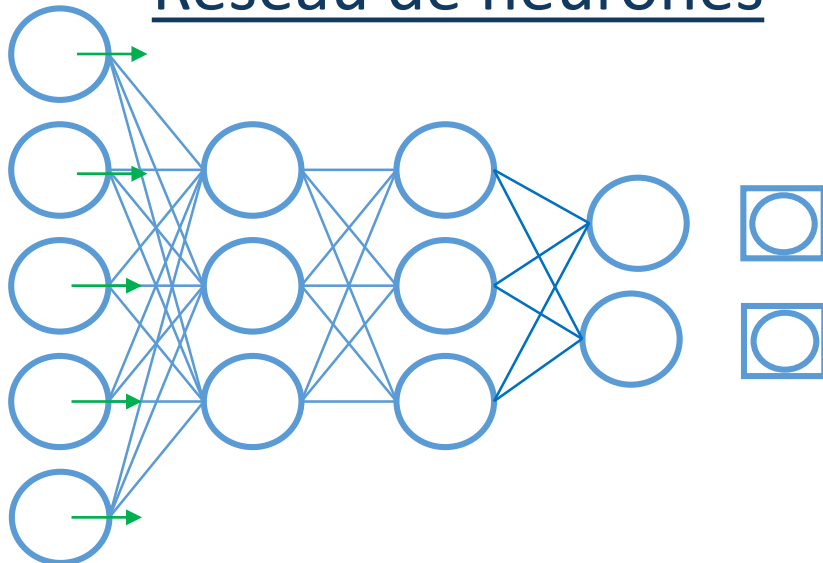
$$E_i^{out} = \text{superviseur}_i^d - sig_i^{out}$$

$$E_i^n = \sum_j w_{ij} \times \delta_j^{n+1}$$

$$W_{ij}^{t+1} = W_{ij}^t + \eta \times sig_i^n \times \delta_j^{n+1}$$

$$\delta^n = sig^n \odot (1 - sig^n) \odot E^n$$

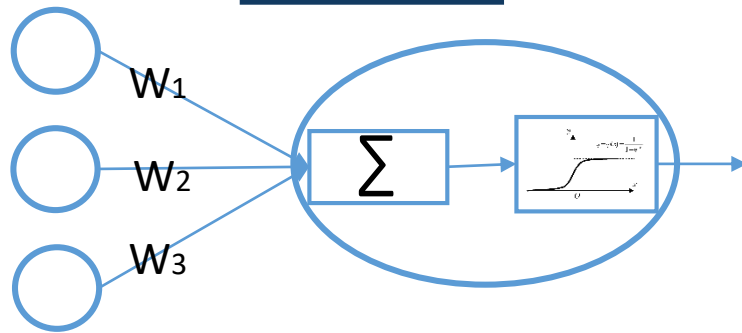
Réseau de neurones



Pierre

Solution 1 : Perceptron multicouches

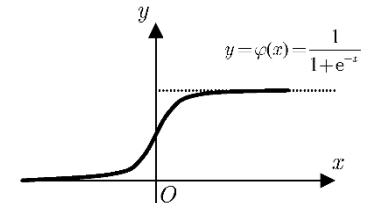
Neurone



Feed forward

$$Pot_j = \sum_i w_{ij} \times x_i$$

$$\text{Signalout} = \text{sigmoid}(Pot)$$



Back propagation

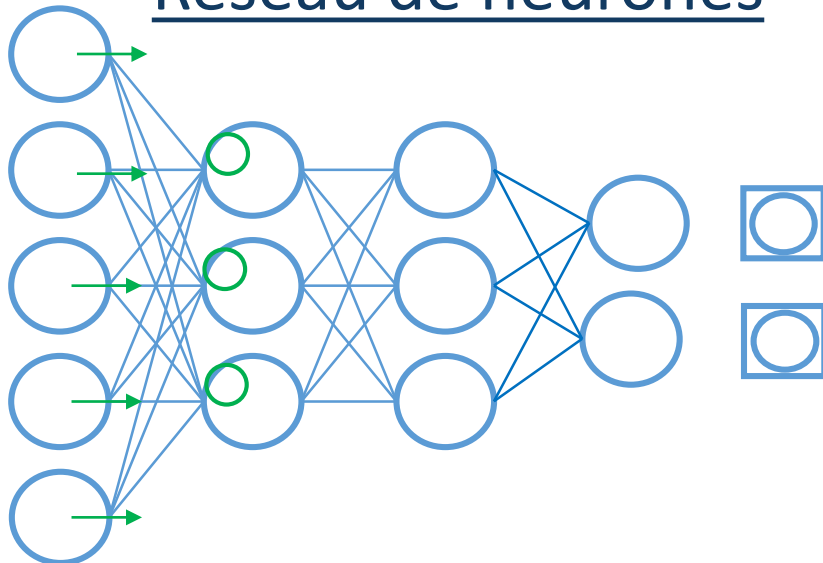
$$E_i^{out} = \text{superviseur}_i^d - \text{sig}_i^{out}$$

$$E_i^n = \sum_j w_{ij} \times \delta_j^{n+1}$$

$$W_{ij}^{t+1} = W_{ij}^t + \eta \times \text{sig}_i^n \times \delta_j^{n+1}$$

$$\delta^n = \text{sig}^n \odot (1 - \text{sig}^n) \odot E^n$$

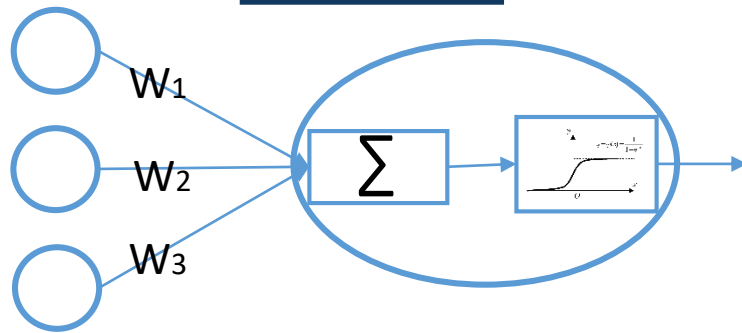
Réseau de neurones



Pierre

Solution 1 : Perceptron multicouches

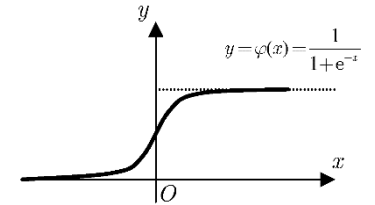
Neurone



Feed forward

$$Pot_j = \sum_i w_{ij} \times x_i$$

$$\text{Signalout} = \text{sigmoid}(Pot)$$



Back propagation

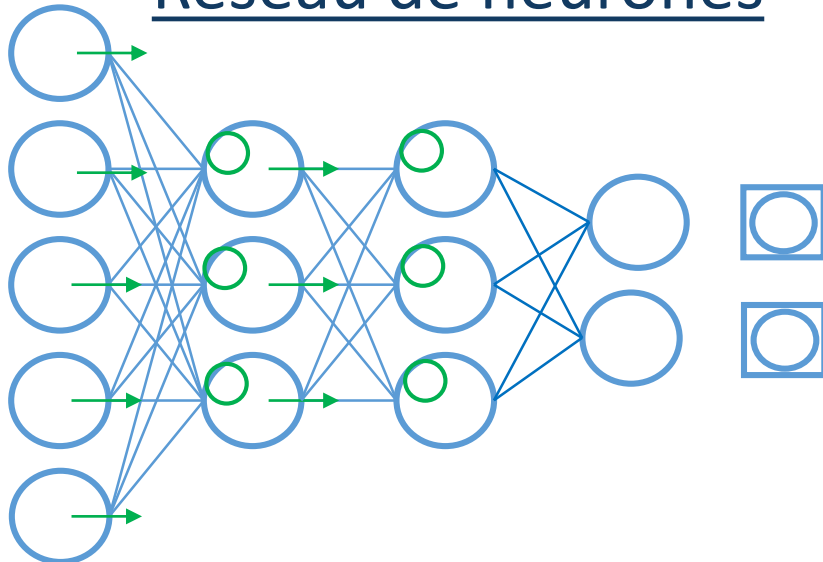
$$E_i^{out} = \text{superviseur}_i^d - \text{sig}_i^{out}$$

$$E_i^n = \sum_j w_{ij} \times \delta_j^{n+1}$$

$$W_{ij}^{t+1} = W_{ij}^t + \eta \times \text{sig}_i^n \times \delta_j^{n+1}$$

$$\delta^n = \text{sig}^n \odot (1 - \text{sig}^n) \odot E^n$$

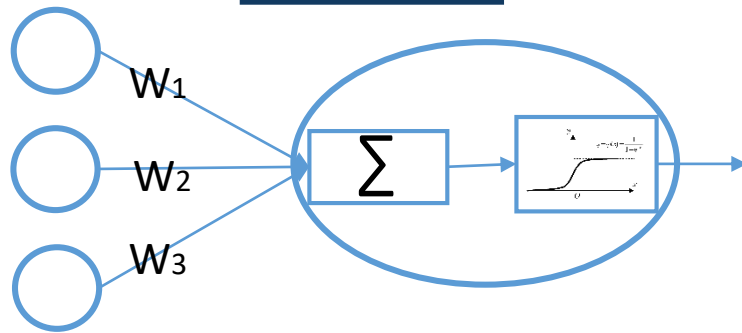
Réseau de neurones



Pierre

Solution 1 : Perceptron multicouches

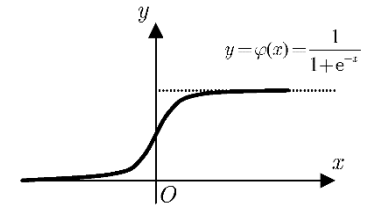
Neurone



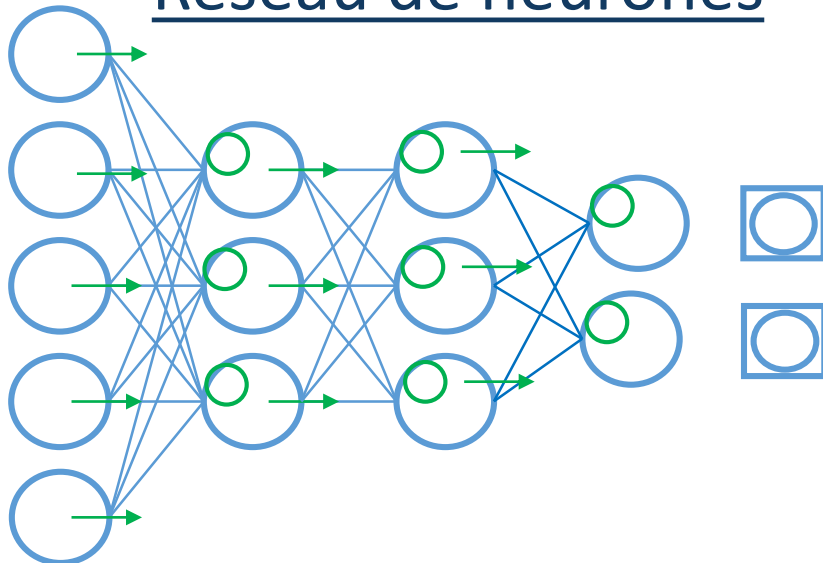
Feed forward

$$Pot_j = \sum_i w_{ij} \times x_i$$

$$Signalout = \text{sigmoid}(Pot)$$



Réseau de neurones



Pierre

Back propagation

$$E_i^{out} = \text{superviseur}_i^d - \text{sig}_i^{out}$$

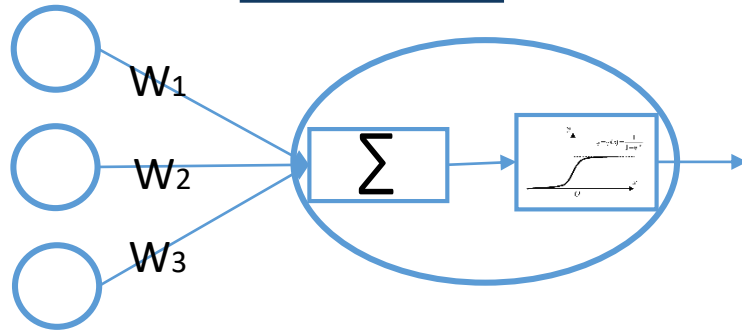
$$E_i^n = \sum_j w_{ij} \times \delta_j^{n+1}$$

$$w_{ij}^{t+1} = w_{ij}^t + \eta \times \text{sig}_i^n \times \delta_j^{n+1}$$

$$\delta^n = \text{sig}^n \odot (1 - \text{sig}^n) \odot E^n$$

Solution 1 : Perceptron multicouches

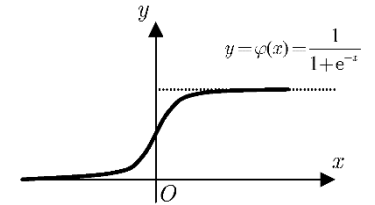
Neurone



Feed forward

$$Pot_j = \sum_i w_{ij} \times x_i$$

$$Signalout = \text{sigmoid}(Pot)$$



Back propagation

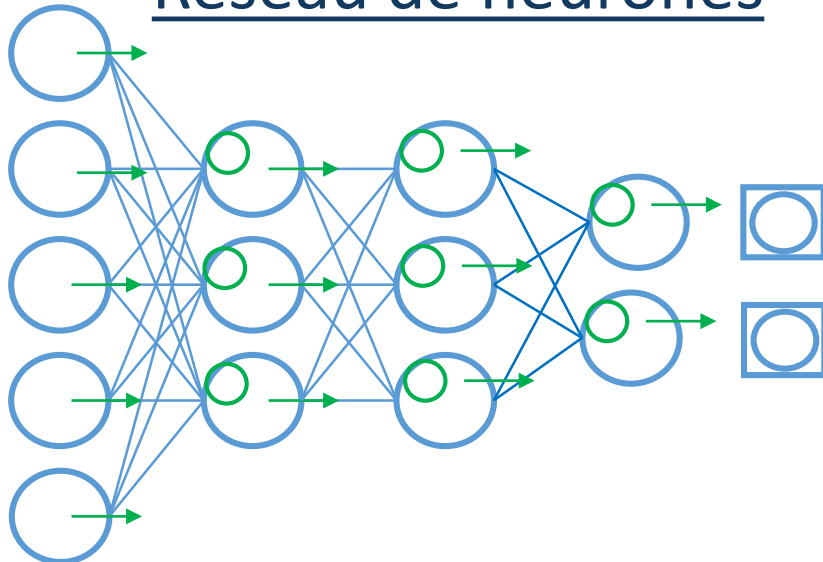
$$E_i^{out} = \text{superviseur}_i^d - sig_i^{out}$$

$$E_i^n = \sum_j w_{ij} \times \delta_j^{n+1}$$

$$w_{ij}^{t+1} = w_{ij}^t + \eta \times sig_i^n \times \delta_j^{n+1}$$

$$\delta^n = sig^n \odot (1 - sig^n) \odot E^n$$

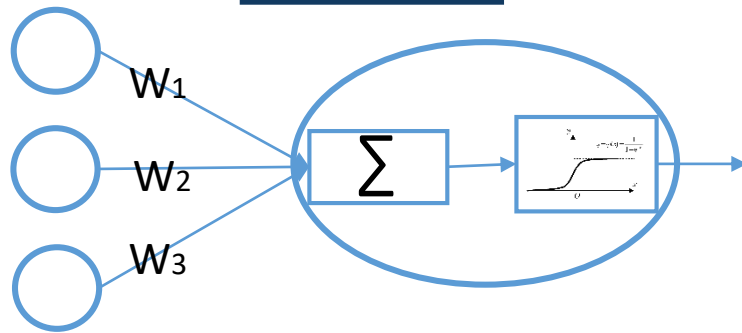
Réseau de neurones



Pierre

Solution 1 : Perceptron multicouches

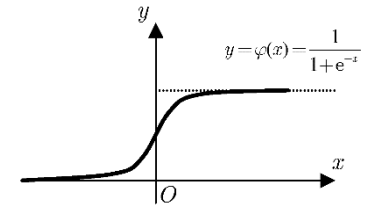
Neurone



Feed forward

$$Pot_j = \sum_i w_{ij} \times x_i$$

$$Signalout = \text{sigmoid}(Pot)$$



Back propagation

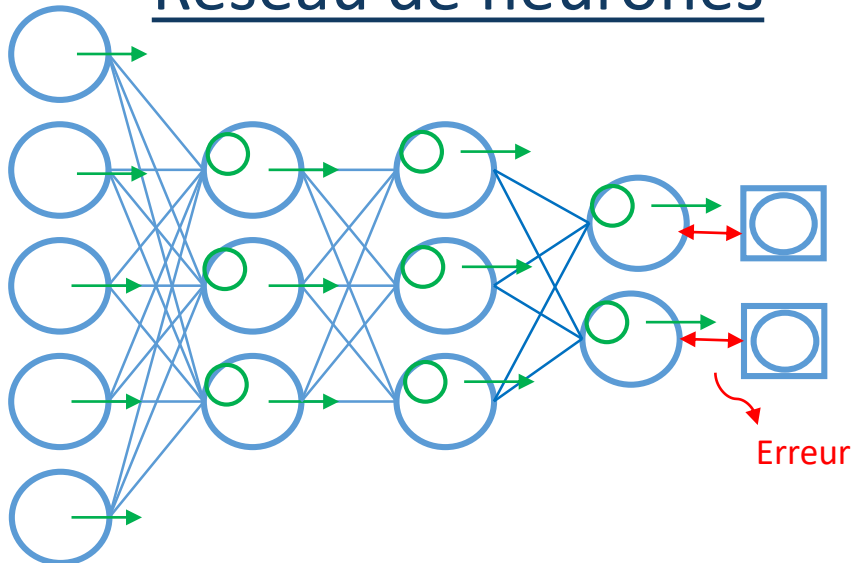
$$E_i^{out} = \text{superviseur}_i^d - sig_i^{out}$$

$$E_i^n = \sum_j w_{ij} \times \delta_j^{n+1}$$

$$W_{ij}^{t+1} = W_{ij}^t + \eta \times sig_i^n \times \delta_j^{n+1}$$

$$\delta^n = sig^n \odot (1 - sig^n) \odot E^n$$

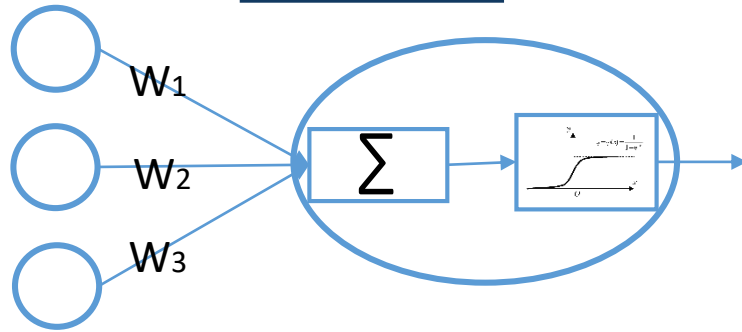
Réseau de neurones



Pierre

Solution 1 : Perceptron multicouches

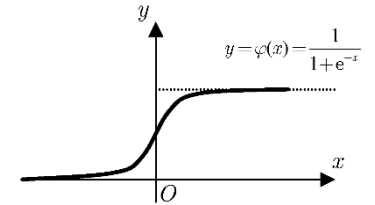
Neurone



Feed forward

$$\text{Pot}_j = \sum_i w_{ij} \times x_i$$

$$\text{Signalout} = \text{sigmoid}(\text{Pot})$$



Back propagation

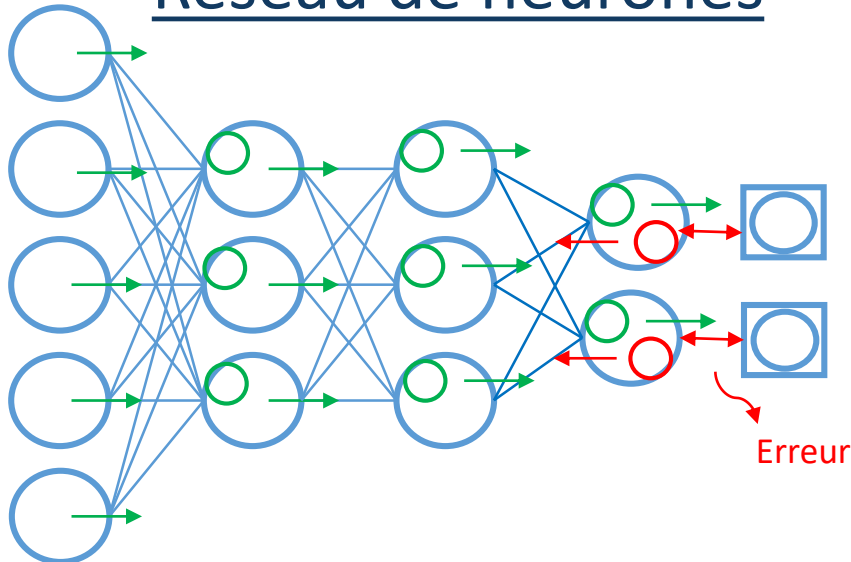
$$E_i^{\text{out}} = \text{superviseur}_i^d - \text{sig}_i^{\text{out}}$$

$$E_i^n = \sum_j w_{ij} \times \delta_j^{n+1}$$

$$w_{ij}^{t+1} = w_{ij}^t + \eta \times \text{sig}_i^n \times \delta_j^{n+1}$$

$$\delta^n = \text{sig}^n \odot (1 - \text{sig}^n) \odot E^n$$

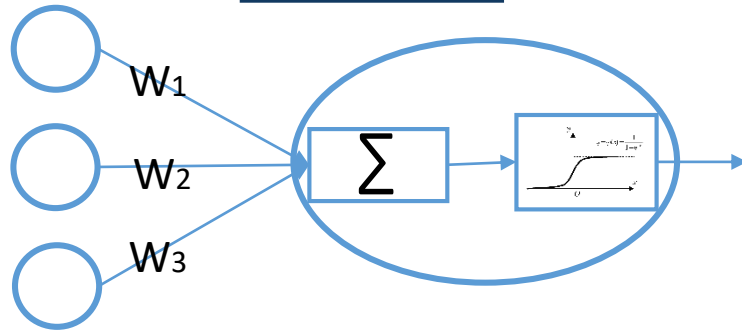
Réseau de neurones



Pierre

Solution 1 : Perceptron multicouches

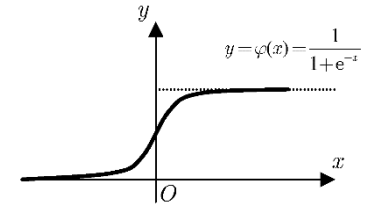
Neurone



Feed forward

$$Pot_j = \sum_i w_{ij} \times x_i$$

$$\text{Signalout} = \text{sigmoid}(Pot)$$



Back propagation

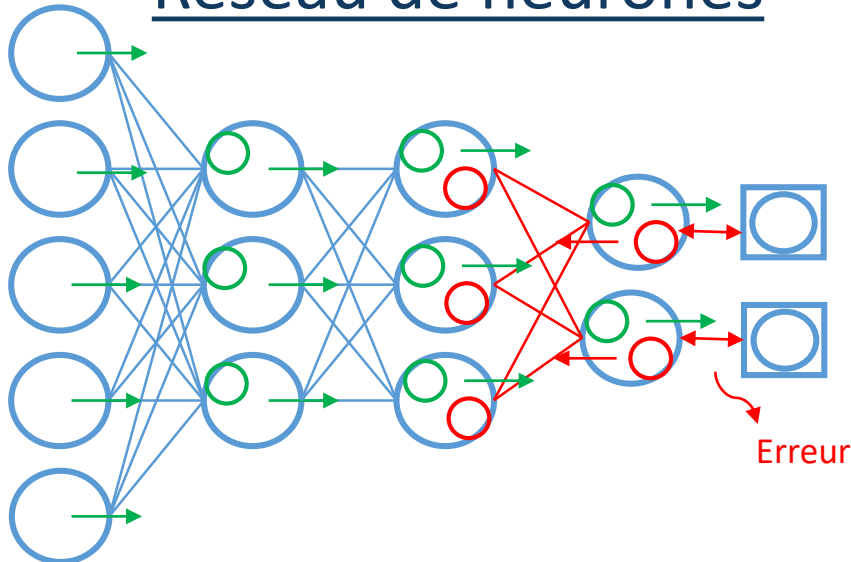
$$E_i^{out} = \text{superviseur}_i^d - \text{sig}_i^{out}$$

$$E_i^n = \sum_j w_{ij} \times \delta_j^{n+1}$$

$$W_{ij}^{t+1} = W_{ij}^t + \eta \times \text{sig}_i^n \times \delta_j^{n+1}$$

$$\delta^n = \text{sig}^n \odot (1 - \text{sig}^n) \odot E^n$$

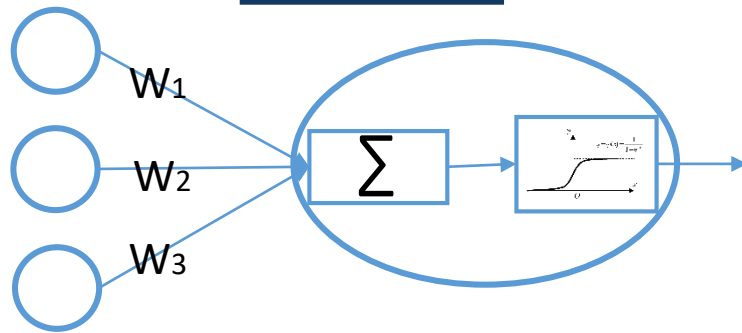
Réseau de neurones



Pierre

Solution 1 : Perceptron multicouches

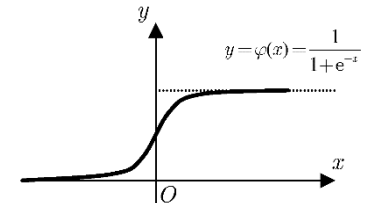
Neurone



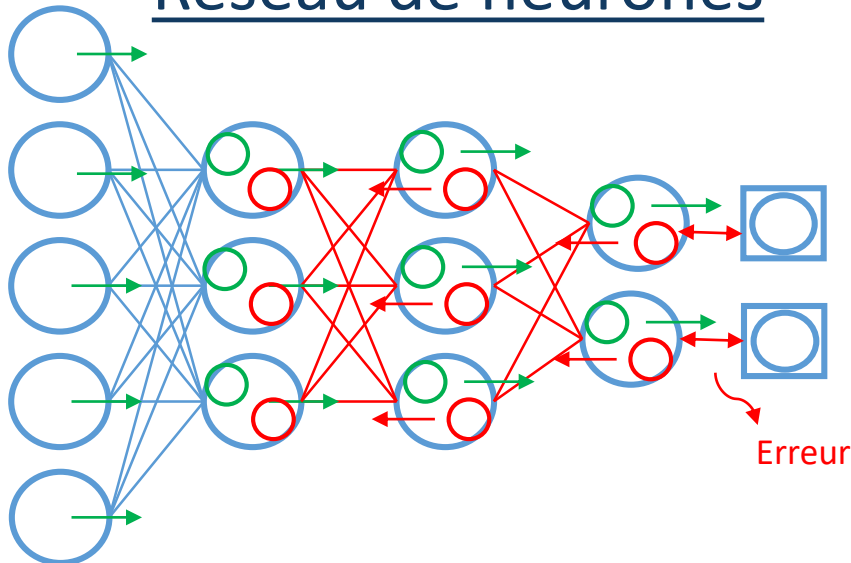
Feed forward

$$Pot_j = \sum_i w_{ij} \times x_i$$

$$Signalout = \text{sigmoid}(Pot)$$



Réseau de neurones



Pierre

Back propagation

$$E_i^{out} = \text{superviseur}_i^d - \text{sig}_i^{out}$$

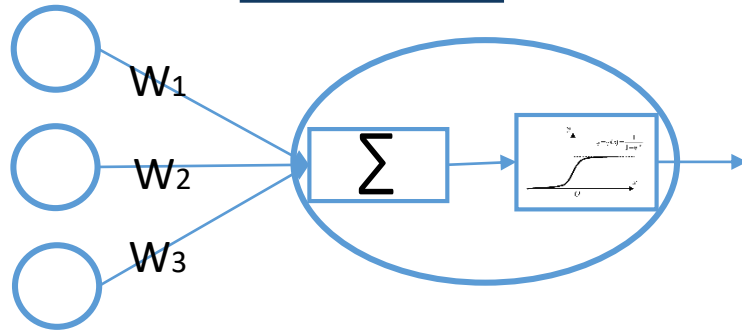
$$E_i^n = \sum_j w_{ij} \times \delta_j^{n+1}$$

$$W_{ij}^{t+1} = W_{ij}^t + \eta \times \text{sig}_i^n \times \delta_j^{n+1}$$

$$\delta^n = \text{sig}^n \odot (1 - \text{sig}^n) \odot E^n$$

Solution 1 : Perceptron multicouches

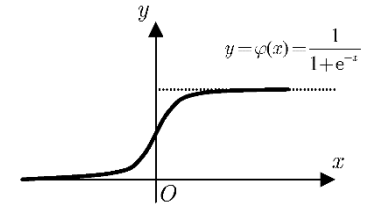
Neurone



Feed forward

$$Pot_j = \sum_i w_{ij} \times x_i$$

$$\text{Signalout} = \text{sigmoid}(Pot)$$



Back propagation

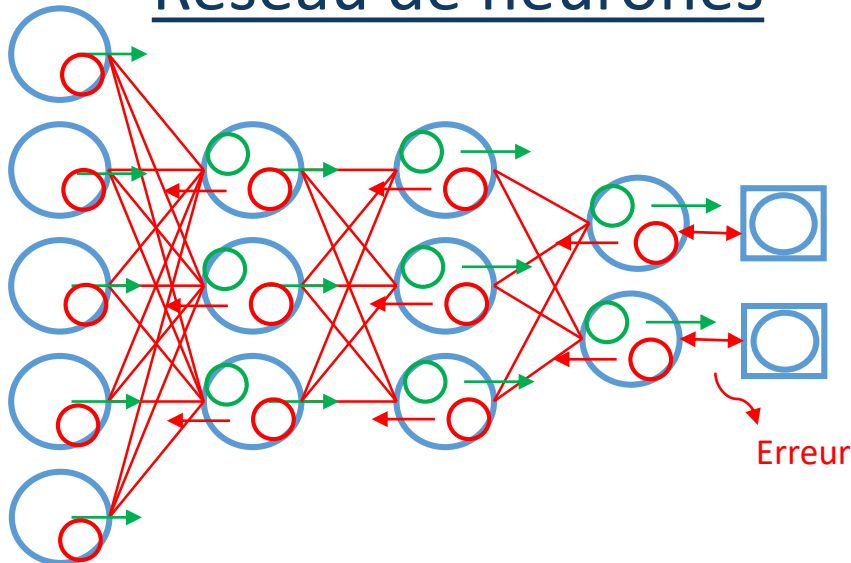
$$E_i^{out} = \text{superviseur}_i^d - \text{sig}_i^{out}$$

$$E_i^n = \sum_j w_{ij} \times \delta_j^{n+1}$$

$$W_{ij}^{t+1} = W_{ij}^t + \eta \times \text{sig}_i^n \times \delta_j^{n+1}$$

$$\delta^n = \text{sig}^n \odot (1 - \text{sig}^n) \odot E^n$$

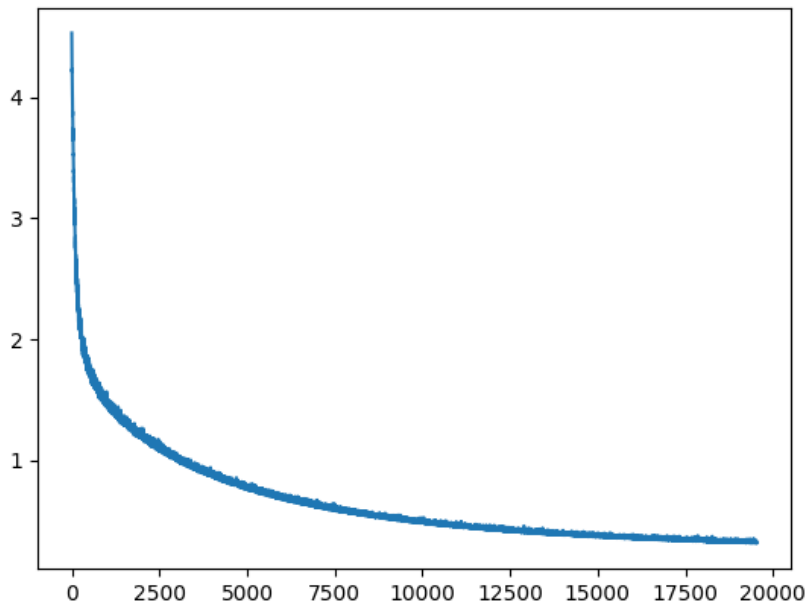
Réseau de neurones



Pierre

Solution 1 : Perceptron multicouches

Coefficients at Penalty:0.01, Samples:5000



Evolution du loss pendant l'entraînement

Nb neurones/couche

10 000

1 024

256

64

17

Total paramètres

10 519 616

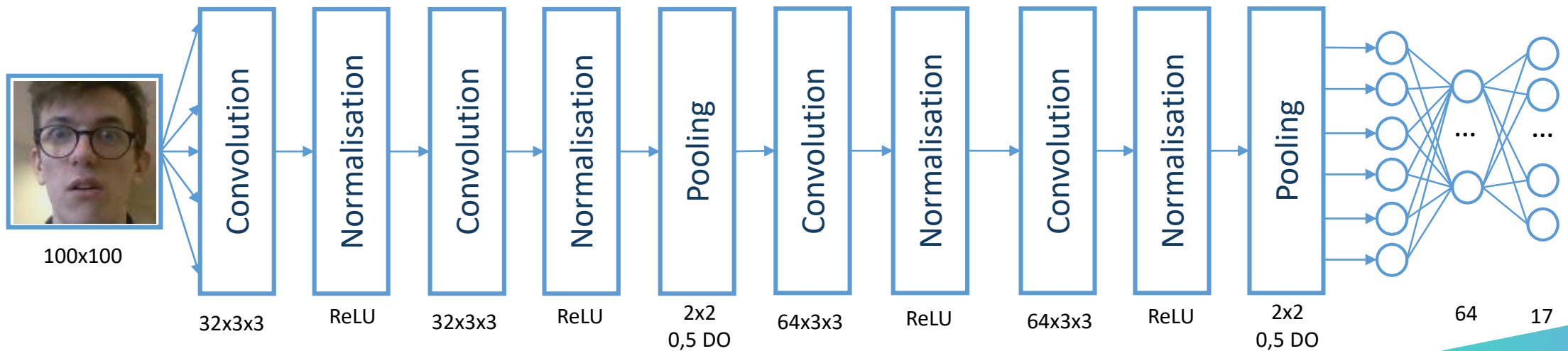
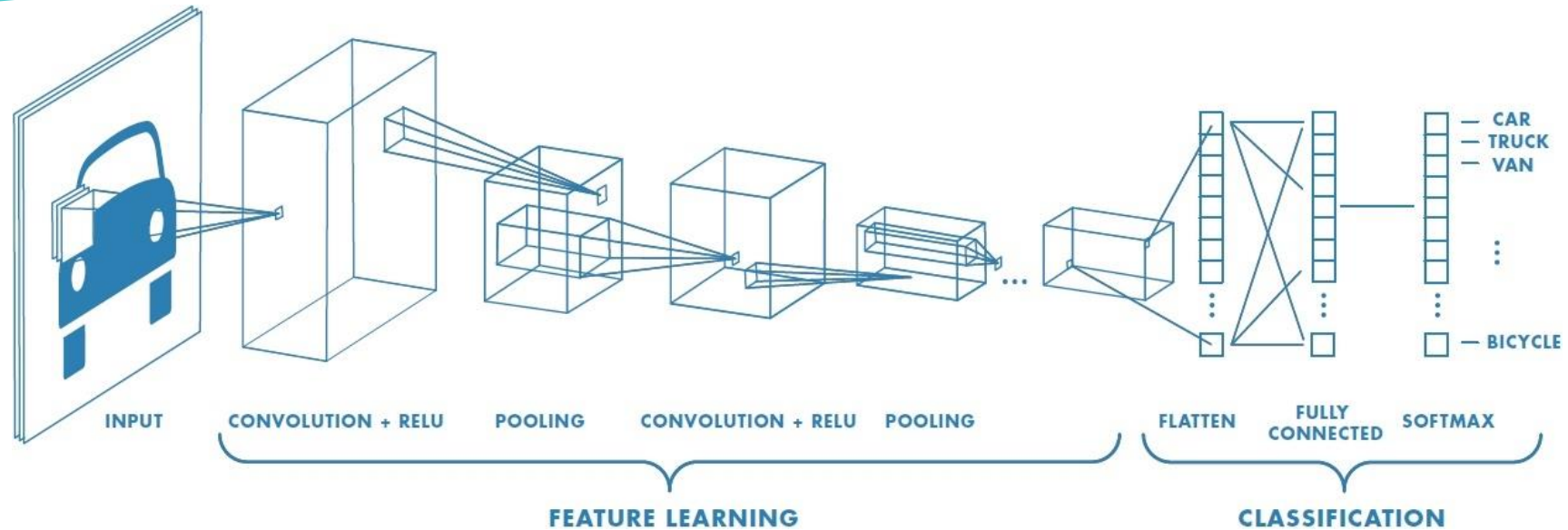
Performances

64,4 %

Entraînement : 00h CPU / 3min24 GPU



Solution 2 : réseau de convolution



Solution 2 : couche de convolution

Convolution

Filtres :

1	-1	-1
-1	1	-1
-1	-1	1

1	-1	1
-1	1	-1
1	-1	1

-1	-1	1
-1	1	-1
1	-1	-1

1	-1	-1
-1	1	-1
-1	-1	1

1	1	-1
1	1	1
-1	1	1

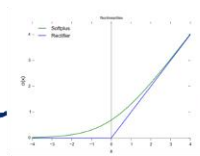
$$(1+1-1+1+1+1-1+1+1)/9 = .55$$

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

Solution 2 : fonction ReLU

ReLU



$\log(\exp(x) + 1)$
 $\max(0, x)$

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.77						

0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

Solution 2 : couche pooling

Pooling

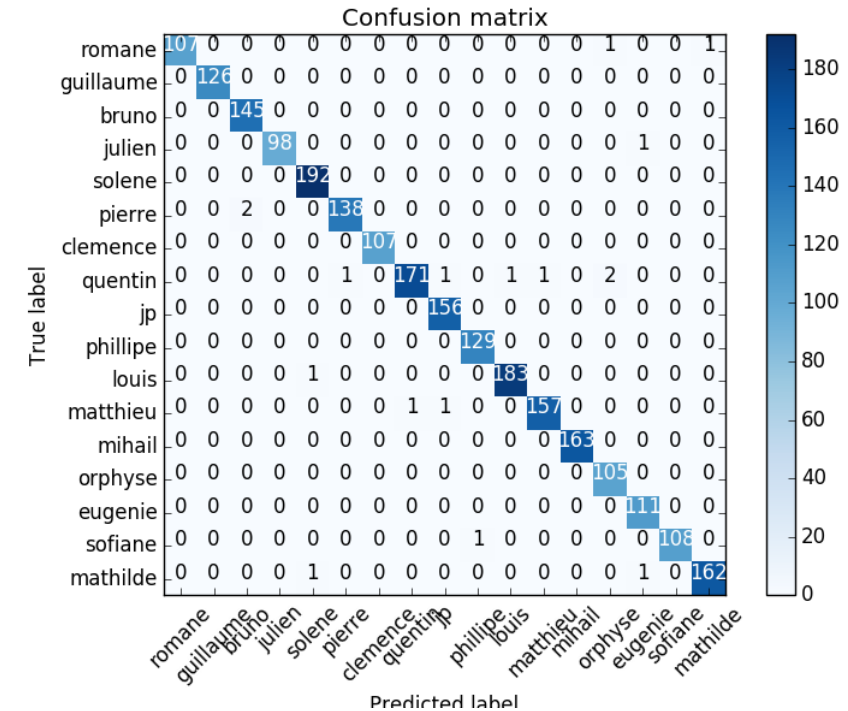
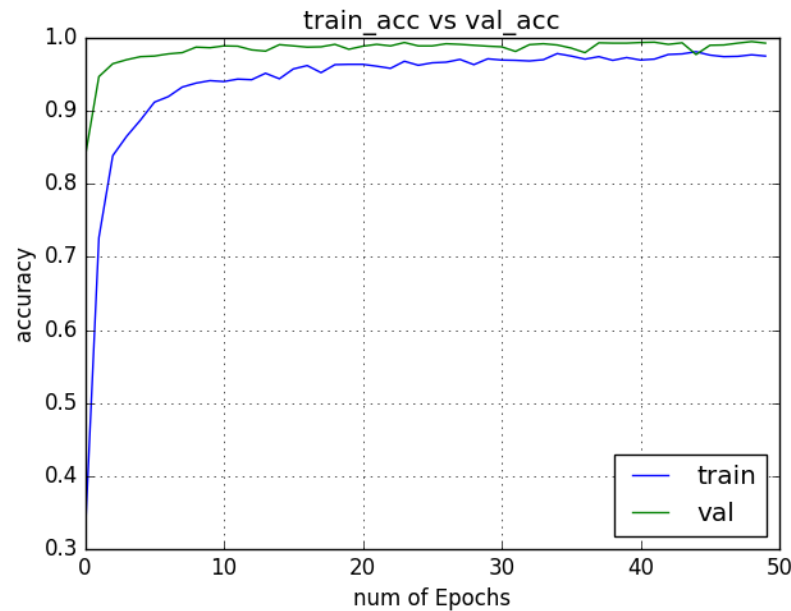
Maximum

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

1.00			

1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

Solution 2 : résultats



Performances

97,1 %

Entrainement : ~5min GPU

Nombre de paramètres : ~9M

DEMO

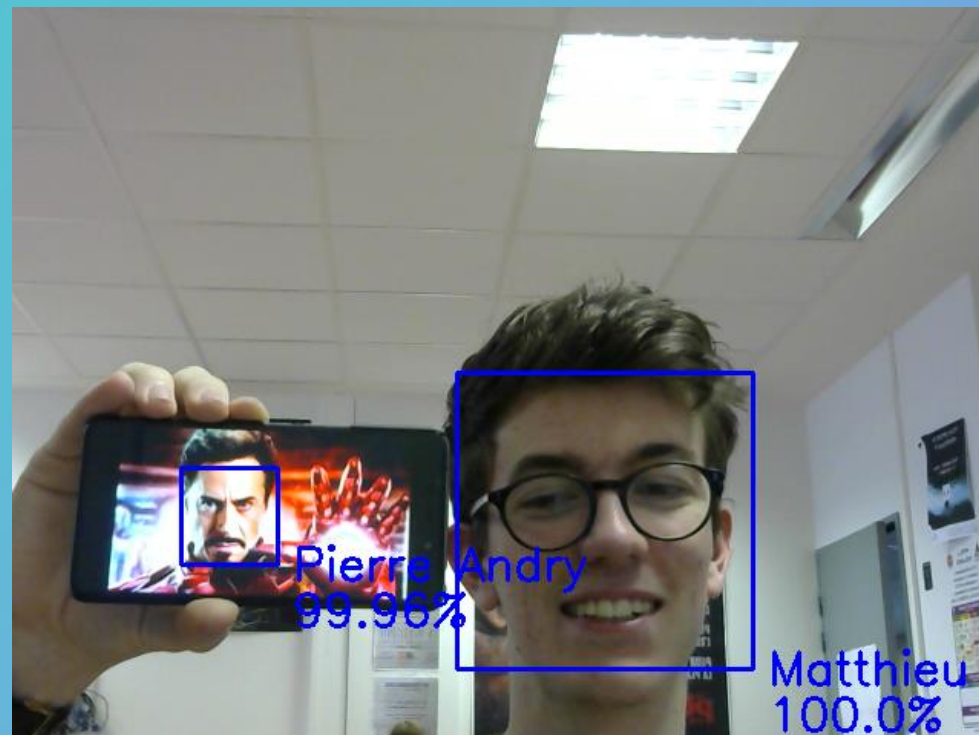


Photo réalisée avec trucage pour la blague

Conclusion reconnaissance

Comment ajouter des nouveaux utilisateurs ?

Comment dimensionner sa base de données ?

Comment dimensionner son model ?

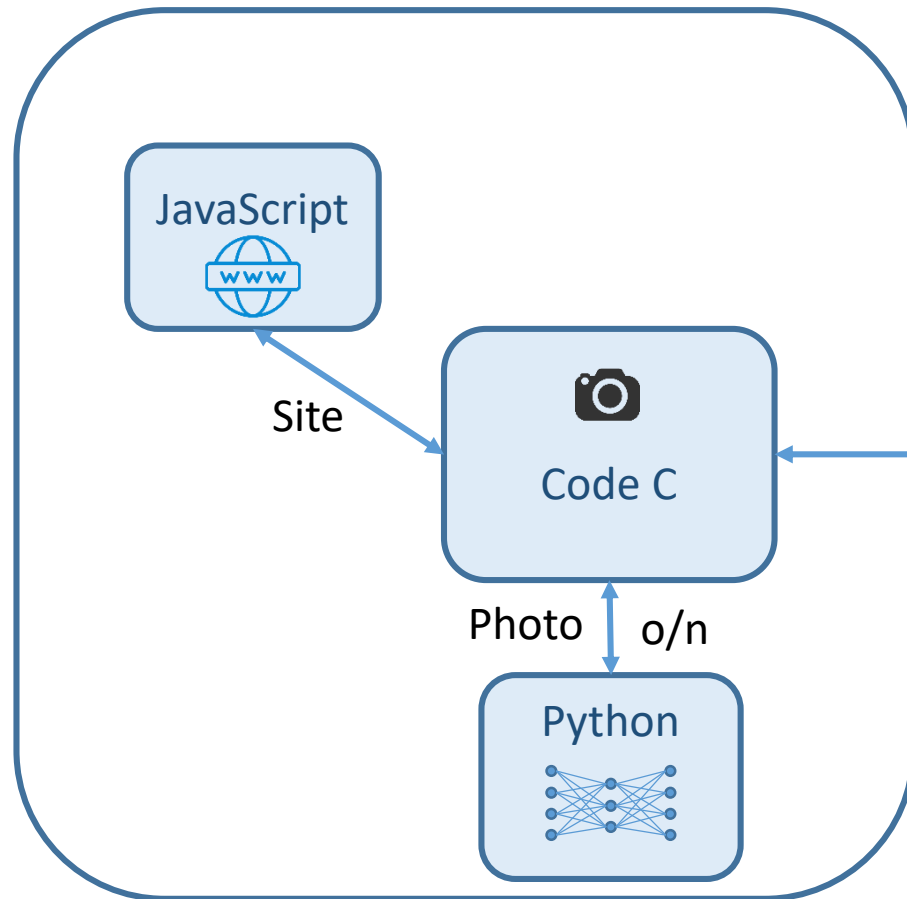


Sécurité

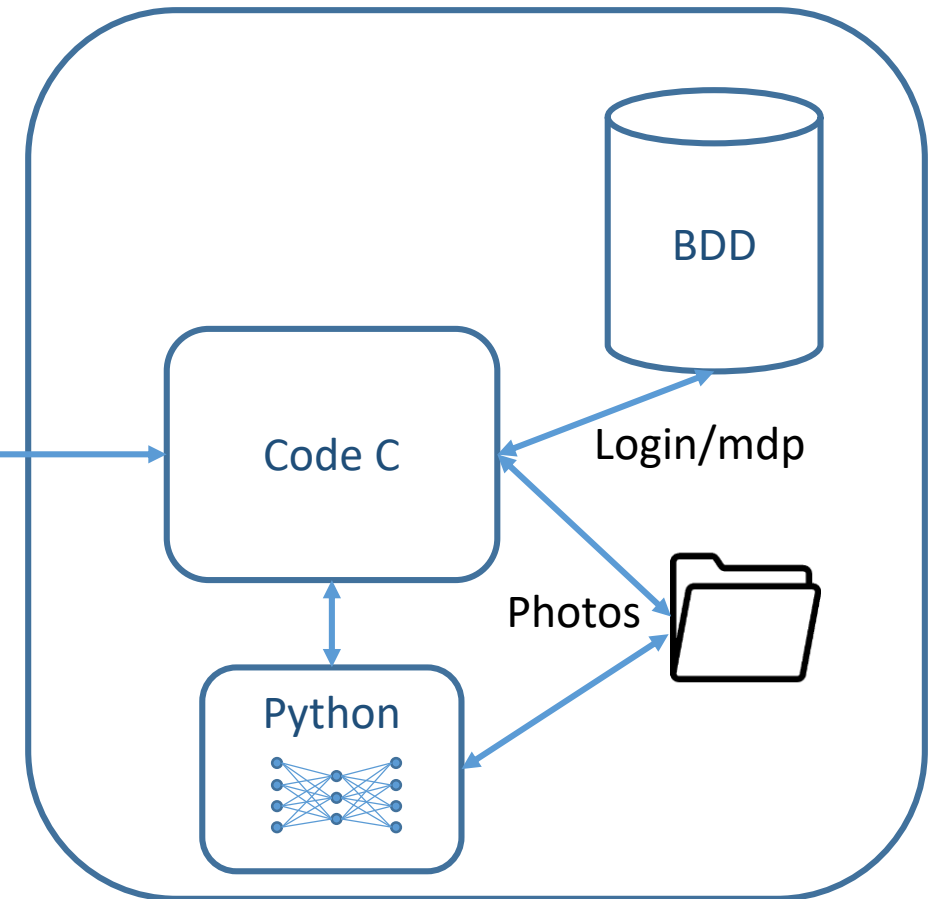
Objectif : sécuriser les données utilisateurs

Sécurité

Client



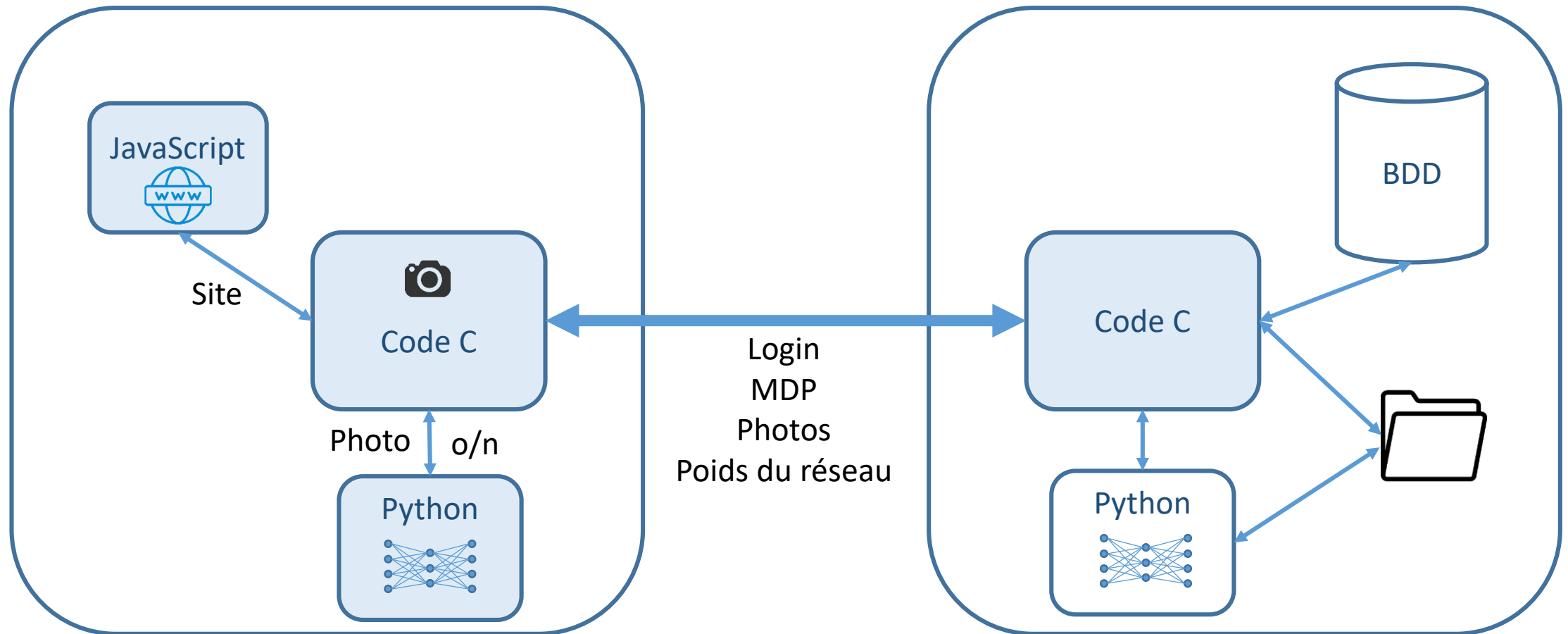
Serveur



Sécurité

Client

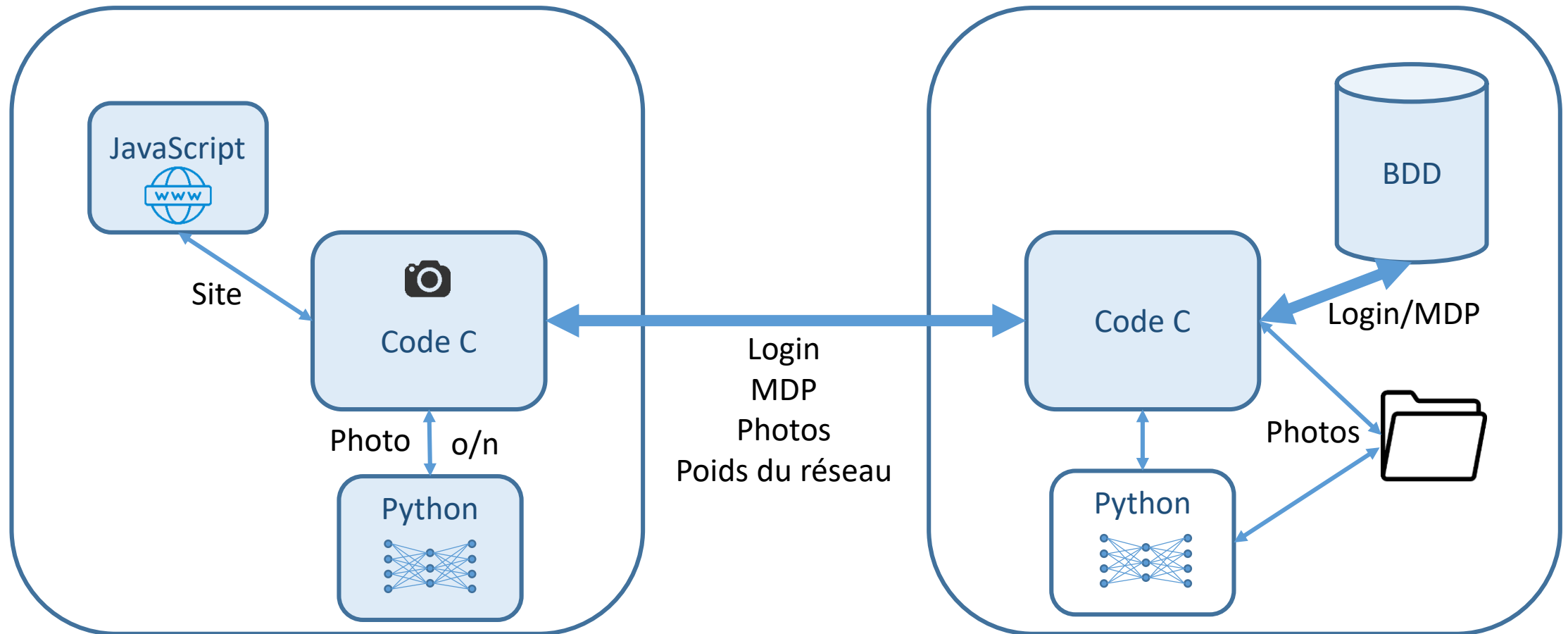
Serveur



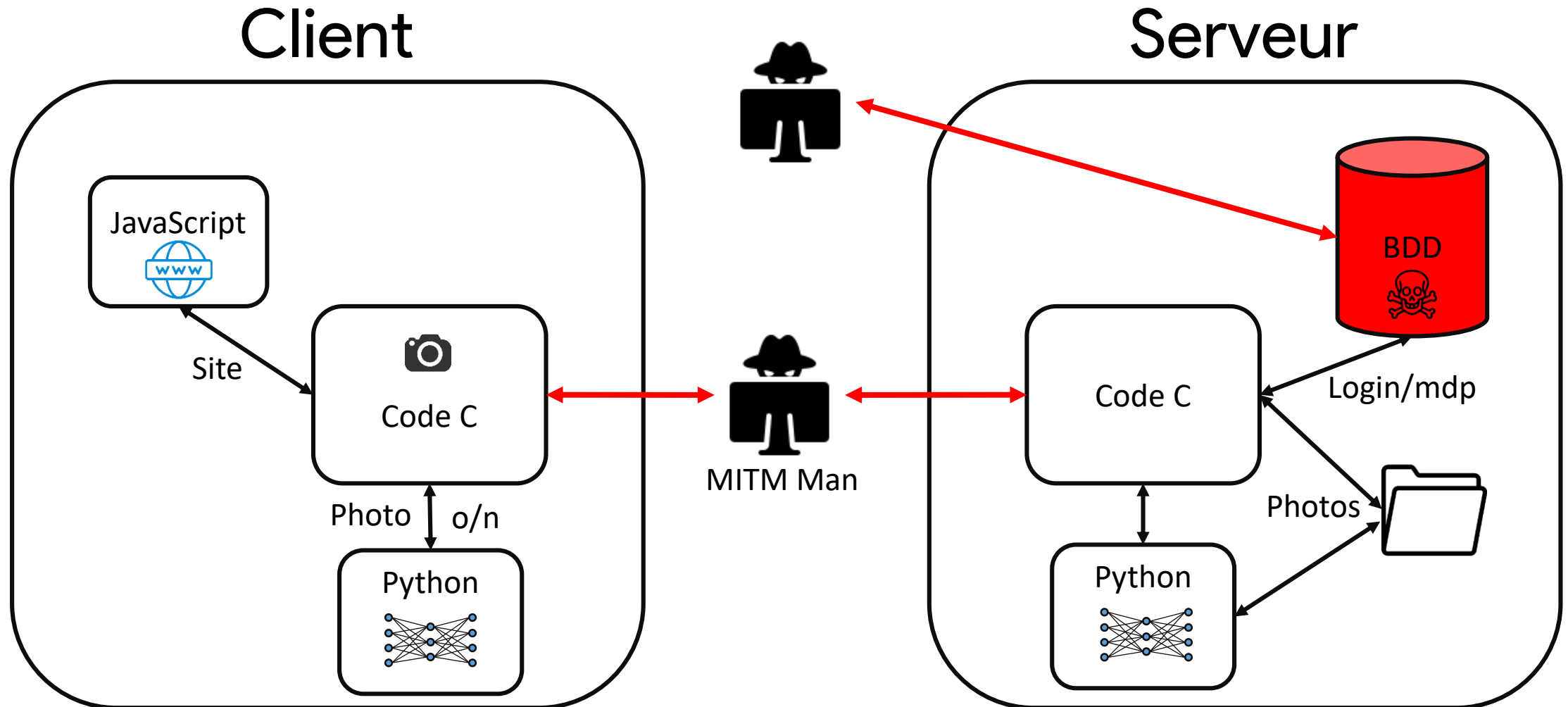
Sécurité

Client

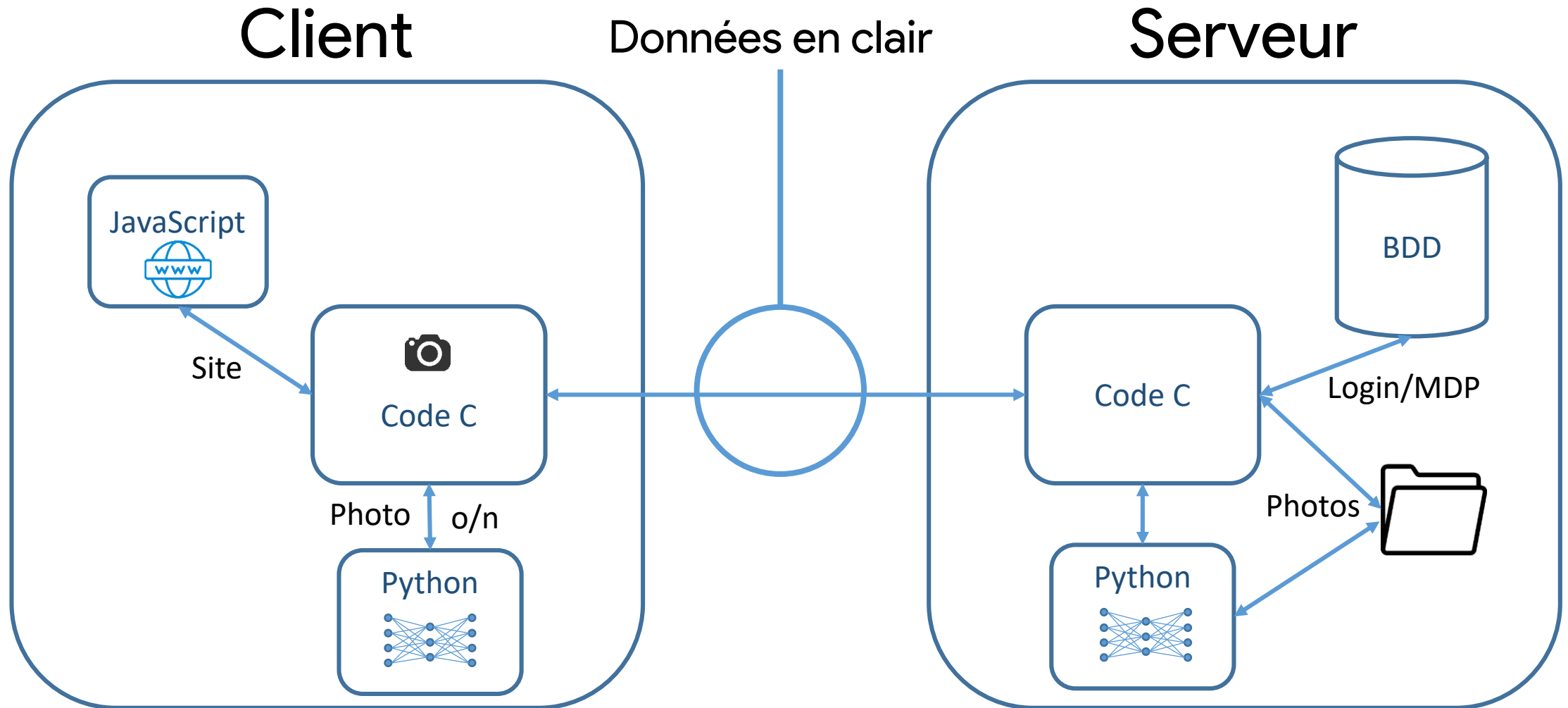
Serveur



Sécurité

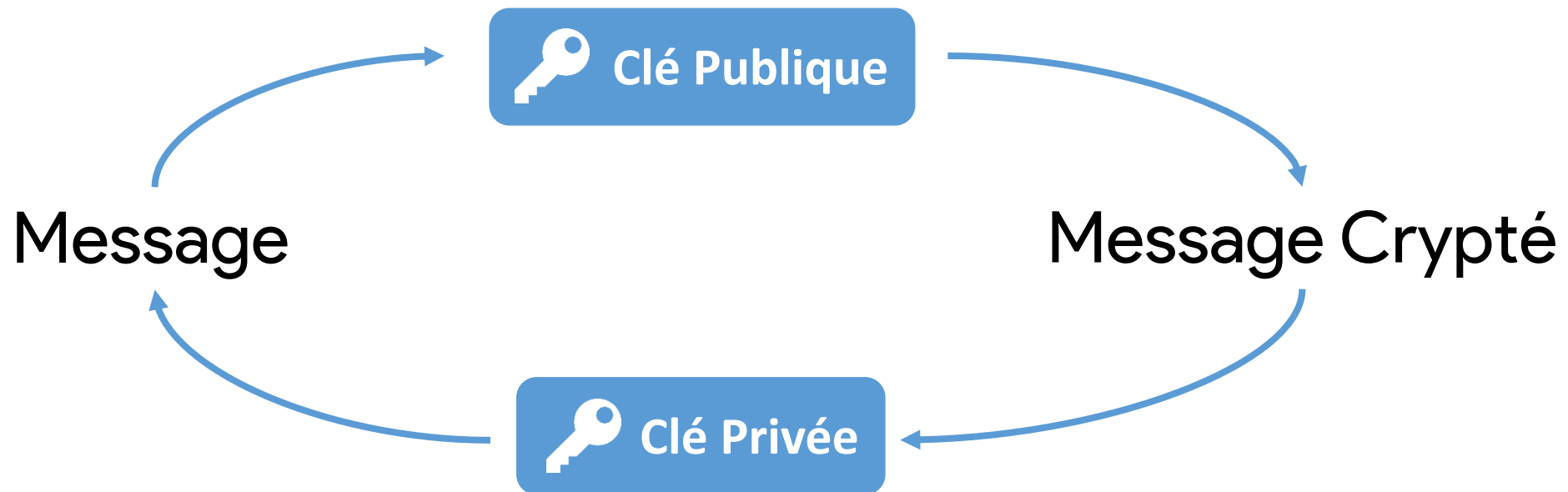


Sécurité



Sécurité: RSA

Chiffrement Asymétrique



Sécurité: RSA

- **Génération de la Clé Privé**

- Choisir p et q , deux nombre premiers, et calculer n

$$n = p \cdot q$$

n sera le modulo et sera présent dans la clé privé ET la clé publique → Choisir p et q assez grands

- Trouver d tel que:

$$\text{pgcd}(d, (p - 1) \cdot (q - 1)) = 1$$

- Nous avons notre clé privée: (d, n) !

Sécurité: RSA

- **Extraction de la clé publique**

- À partir de la clé privée (d, n) , nous devons trouver e tel que:

$$e \cdot d \equiv 1 \pmod{(p-1) \cdot (q-1)}$$

Notre clé publique est donc (e, n) !

- **Cryptage et Décryptage**

- Pour un message M , nous avons le message crypté C :

$$C \equiv M^e \pmod{n}$$

- Pour un message crypté C , nous avons le message M :

$$M \equiv C^d \pmod{n}$$

Sécurité: RSA

Client

Génération des clés
privée et publique

Serveur

Génération des clés
privée et publique

Sécurité: RSA

Client

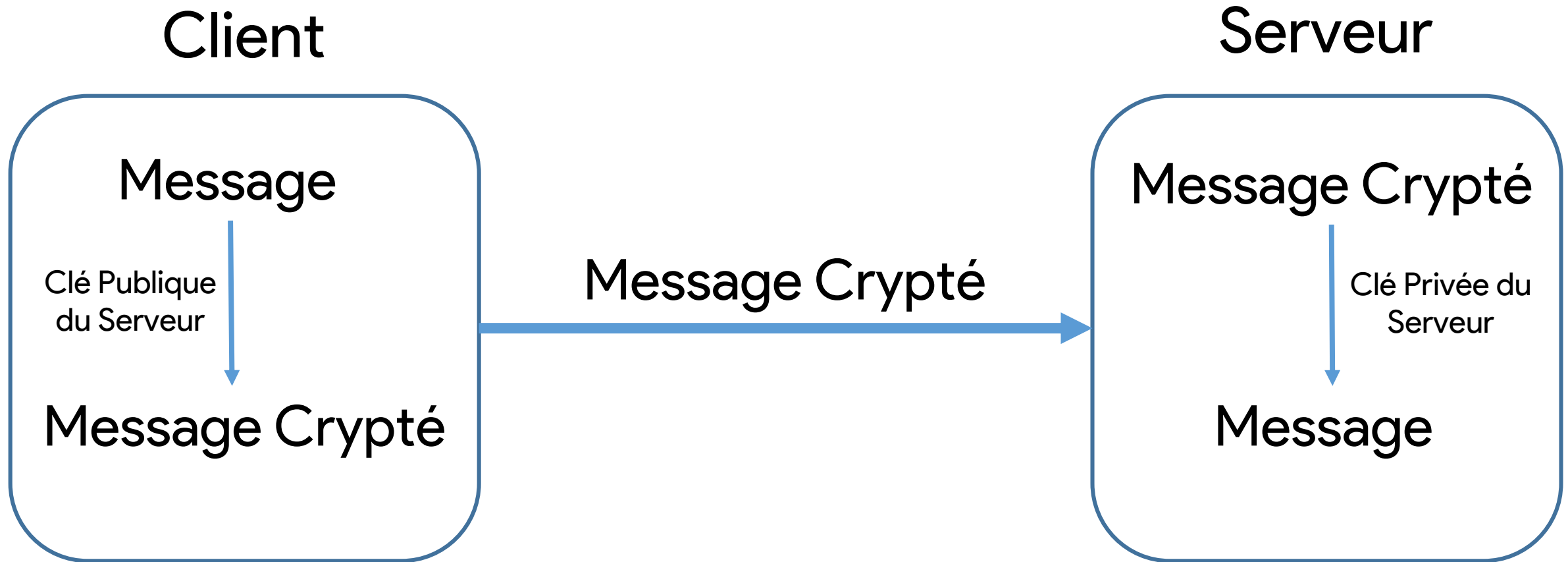
Réception clé
publique du Serveur

Échange des clés publiques

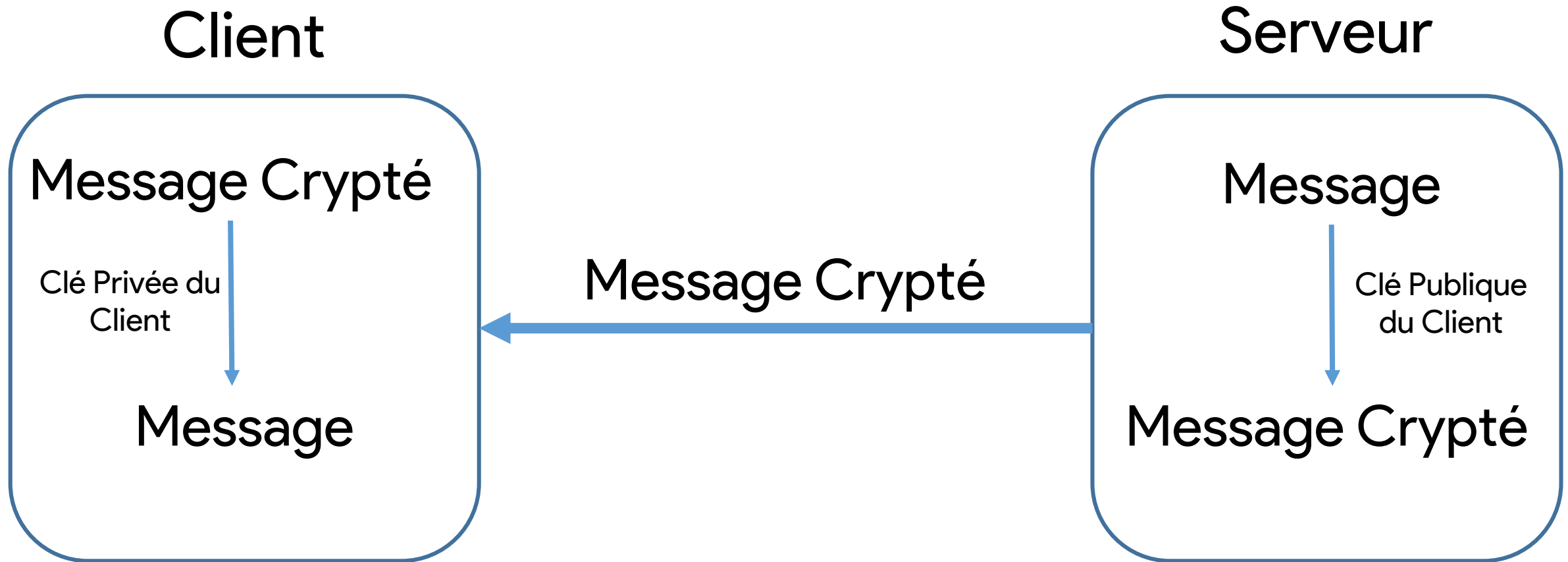
Serveur

Réception clé
publique du Client

Sécurité: RSA



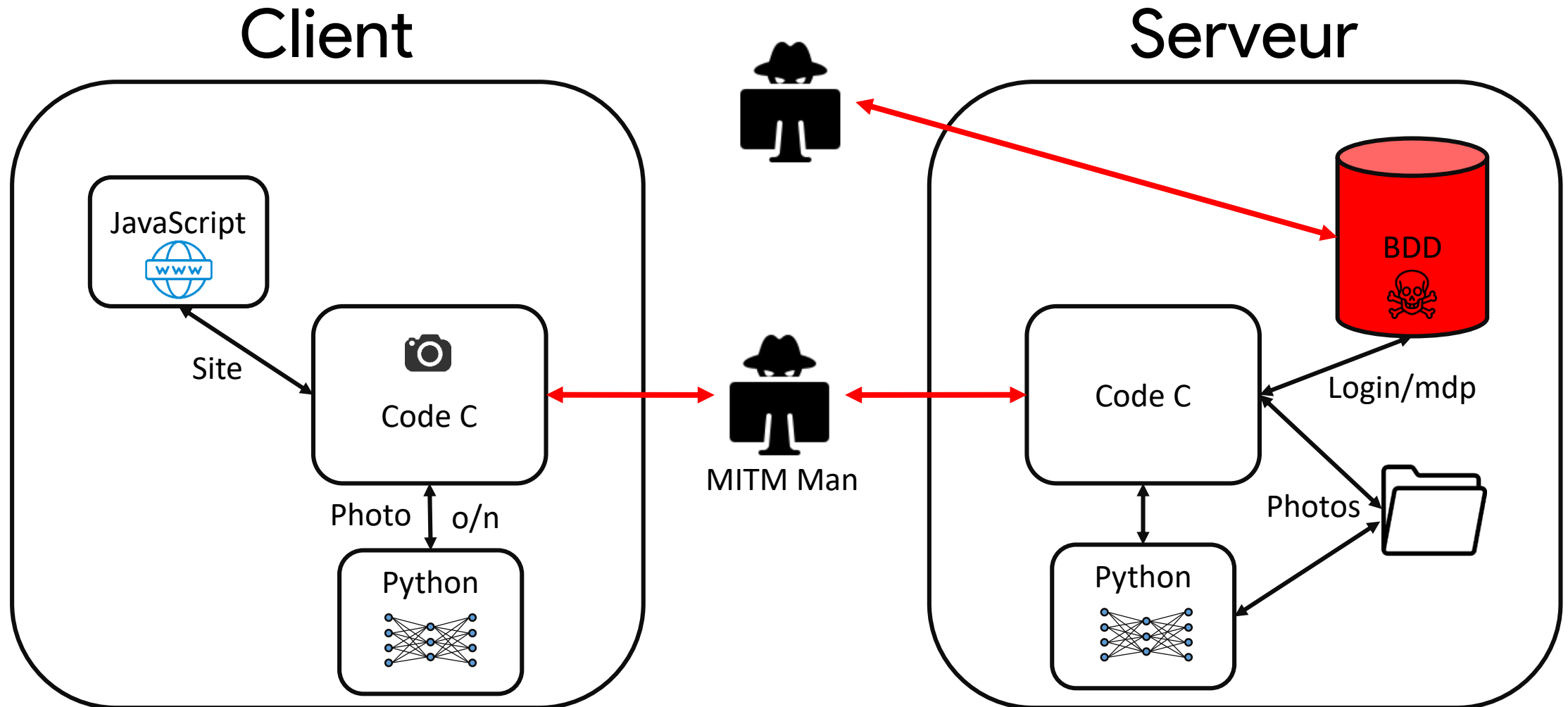
Sécurité: RSA



Sécurité : RSA

- ✓ Génération des clés
- ✓ Chargement des clés
- ✓ Cryptage d'un message
- ✓ Décryptage d'un message
- ✗ Transmission du message entre le client et le serveur

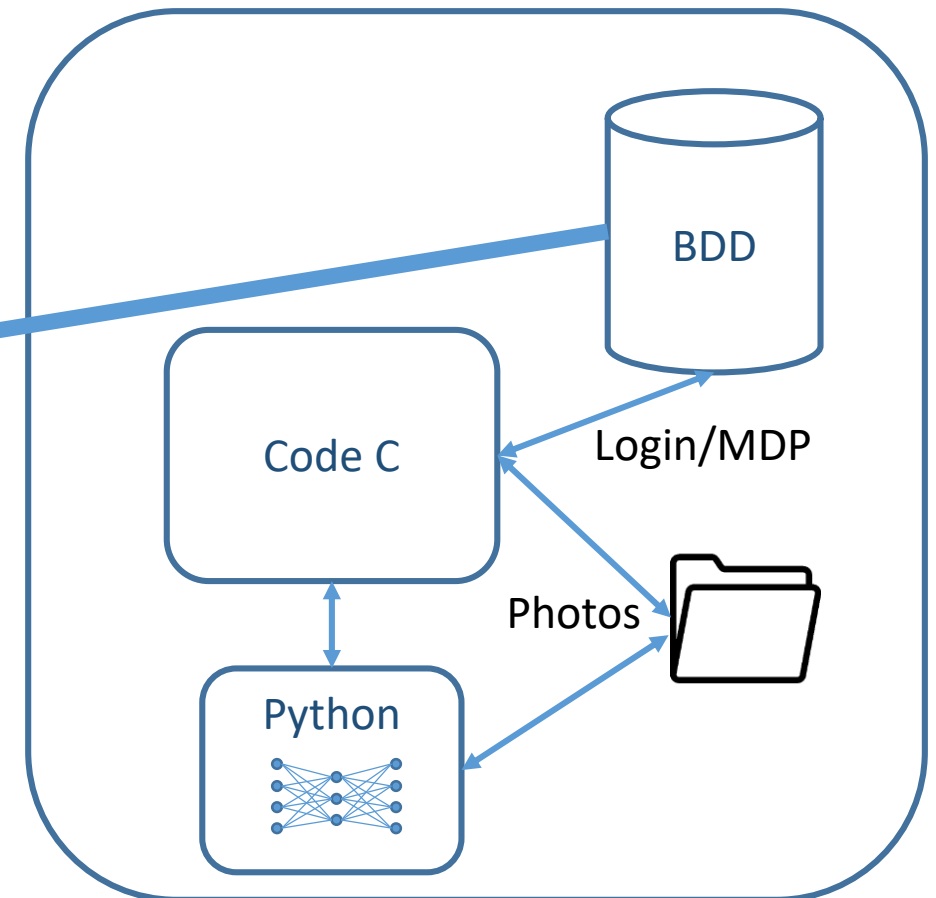
Sécurité



Sécurité

ID User	Login	Password	...
1	harry.covert@gmail.com	password	...
2	toto@hotmail.fr	123456	...
3	louis.lharidon@etu.u-cergy.fr	azerty	...
4	dipolisplayer@gmail.com	azertyuiop	...
5	contact@gamys.com	qsdghjklm	...
...

Serveur



Sécurité : Hachage

« password » $\xrightarrow{\text{MD5}}$ 5f4dcc3b5aa765d61d8327deb882cf99

- **Irréversible:**

5f4dcc3b5aa765d61d8327deb882cf99 $\xrightarrow{\text{X}}$ « password »

- **Empreinte numérique:**

« password » = « password » \rightarrow 5f4dcc3b5... = 5f4dcc3b5...

« password » \neq « Password » \rightarrow 5f4dcc3b5... \neq dc647eb65...

- Attaque par force brute
- Attaque par dictionnaire
- Attaque par « Rainbow Table »

Sécurité : Hachage salé

- Principe: Concaténer une chaîne de caractère au mot de passe

password + Sel1 $\xrightarrow{\text{MD5}}$ d0018b54a667dcc19a1377823f3f8c65

password + Sel2 $\xrightarrow{\text{MD5}}$ fe7e026084f426948798693f19e9cd19

- **Un Sel Unique par utilisateur**

password = password \rightarrow d0018b54a667d... \neq fe7e026084f426...

Sécurité: Hachage Implémentation

Client

Récupération du
mot de passe de
l'utilisateur

Serveur

Attente

Sécurité: Hachage Implémentation

Client

Récupération du
mot de passe de
l'utilisateur

Envoie du mot de passe

Serveur

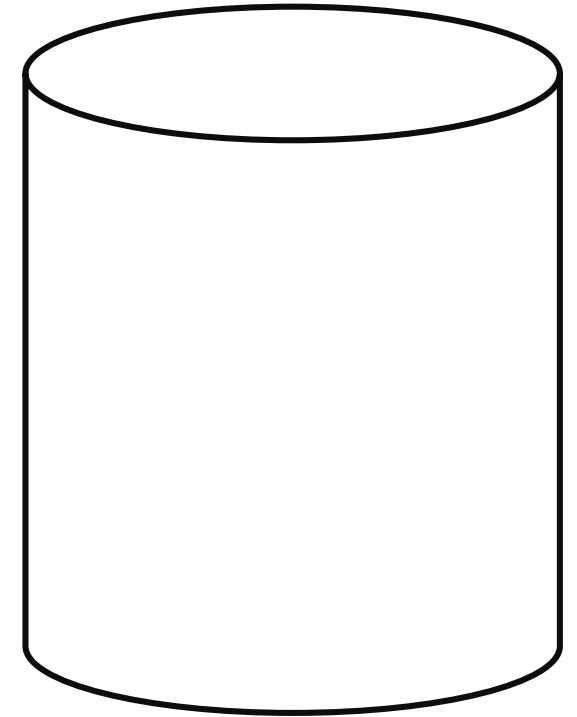
Récupération du
mot de passe

Sécurité: Hachage Implémentation

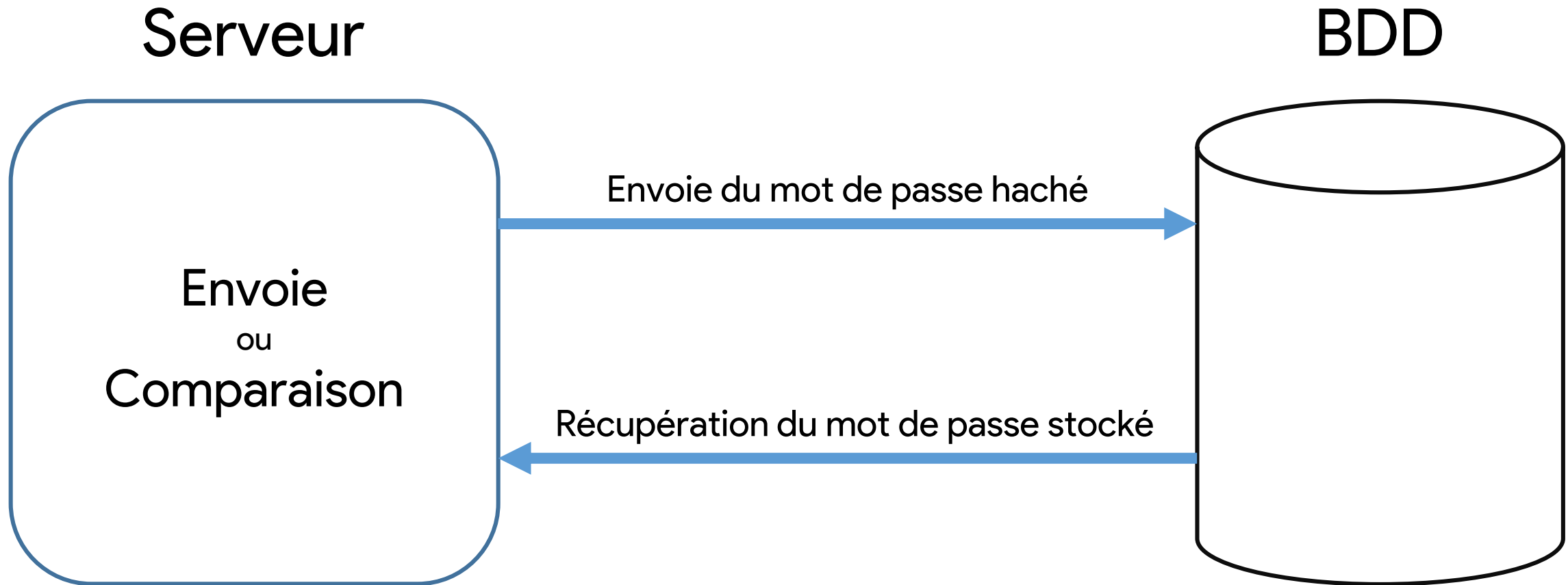
Serveur

Hachage du mot de
passe

BDD



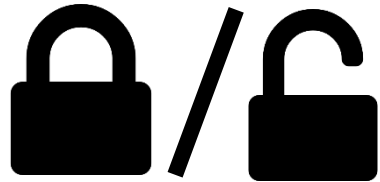
Sécurité: Hachage Implémentation



Sécurité : Conclusion



Mot de passe Face Key

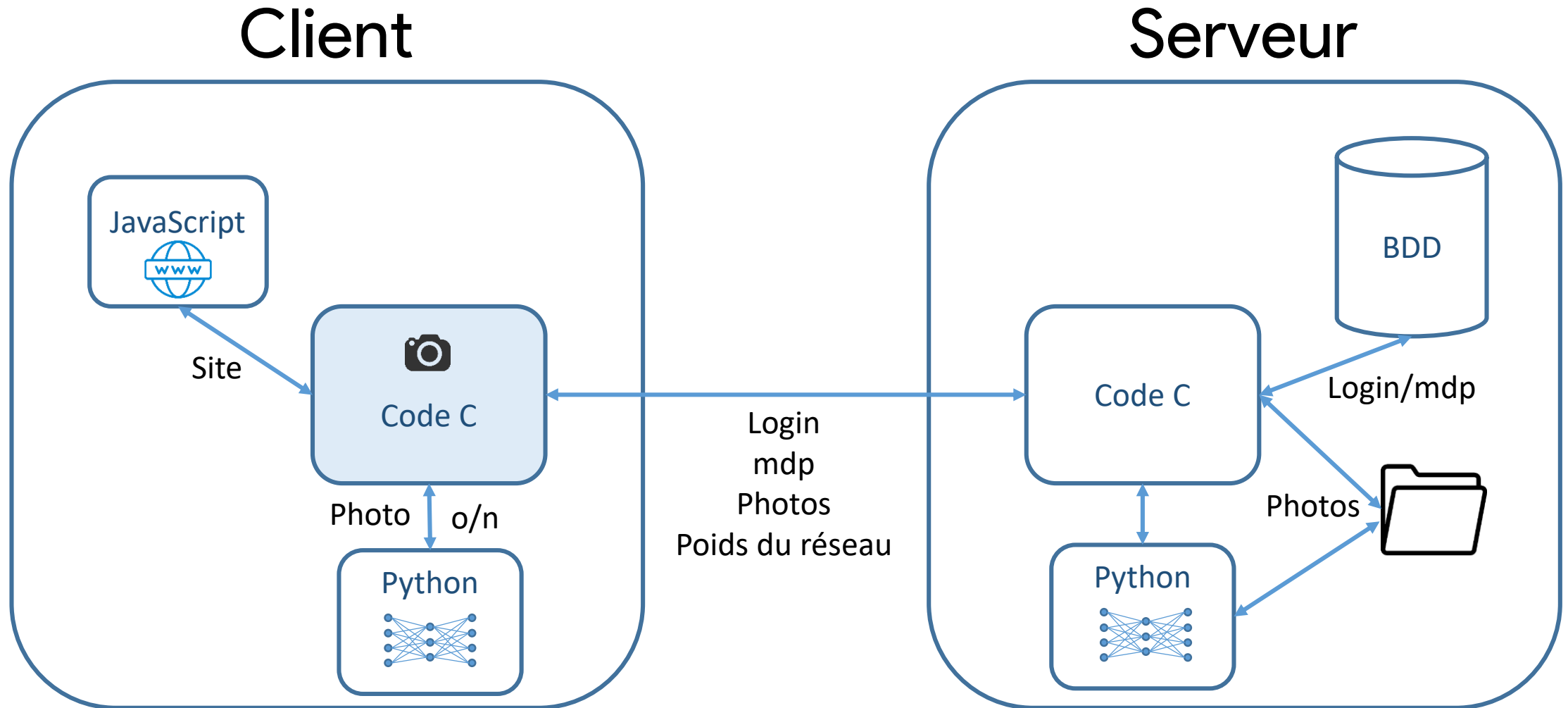


Communication Client/Serveur



Combinaisons Login/MDP

Conclusion



Conclusion : Où en sommes nous ?

Fonctionnelle

Réseau client/serveur
Base de données
Application mobile

Apprentissage
Localisation visage
Reconnaissance visage

Hashage

En cours

Plug-in communication
UDP
Interactions utilisateur

Refonte de la base de
données de visages

RSA

Non implémenté

Récolte statistiques
utilisateur

Plug-in gestion compte
Ajout de compte
utilisateur

Algorithme de
localisation HOG

Hashage salé

Conclusion : Améliorations possibles

Interface graphique de l'application

Regroupement de la partie client dans le plug-in

Reconnaissance des émotions pour statistiques

Sécurisation des combinaison login/mdp pour les sites

Maths pour l'informatique

Traitement d'image

Algèbre linéaire

Recherche Bibliographique

Programmation système

Base de données

Gestion de projet
informatique

Réseau

Développement web

Développement Android

Génie logiciel



Face Key

Merci !