

# Projet réseau PEI-2 2025-2026

PROTOCOLES RÉSEAUX IP, SQL & HTML  
(20h)

## 1. OBJECTIFS DU PROJET

Ce projet a pour objectif de vous faire manipuler des notions de réseaux et de programmation abordées lors de vos TP. Vous serez ainsi amenés à récupérer des informations depuis un serveur UDP (*Client/Serveur Python*), à traiter et analyser les données reçues en les comparant à des données stockées en bases de données (SQL) pour enfin afficher des informations dans une IHM (*HTML*).

## 2. DESCRIPTION DU PROJET

Pour ce projet, vous allez être au cœur de la gestion de commandes de pizzas. Votre objectif est de valider les commandes de clients ; en fonction de la disponibilité des moyens de productions et des délais de fabrication et de livraison. Vous serez ensuite amenés à générer des statistiques de ventes et des indicateurs de performance de la production.

Pour cela, un serveur UDP reçoit et concentre les commandes de pizzas de clients. Les informations concernant les utilisateurs, les pizzas et les capacités des moyens de production sont stockées dans une base de données SQL.

Votre travail sera dans un premier temps, de valider la faisabilité des commandes clients en fonction de l'heure de livraison souhaitée et du temps de livraison. Dans un second temps, vous gèrerez les capacités et les disponibilités des moyens de production. Enfin, vous afficherez des informations de production et des statistiques de ventes sur un dashboard (IHM).

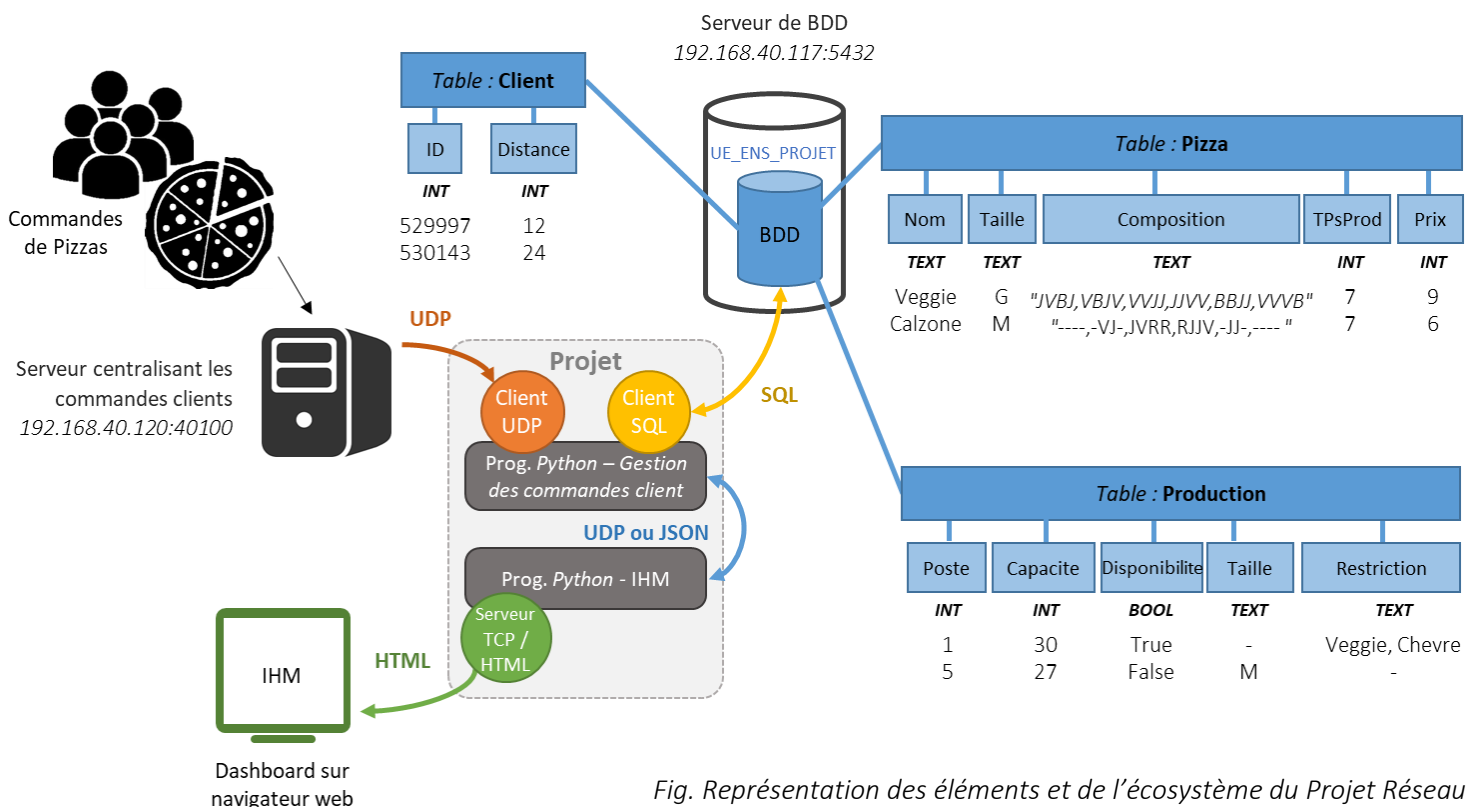


Fig. Représentation des éléments et de l'écosystème du Projet Réseau

## DÉROULÉ DU PROJET

Le *Projet Réseau* aura une durée de 20h et se déroulera sur 5 séances.

Attention : ceci n'est pas un « Méga-TP », il s'agit bien d'un projet ; c'est-à-dire que votre notation prendra tout autant en compte votre gestion de projet que vos rendus. Ces derniers devront comprendre un rapport détaillant votre méthodologie, les outils utilisés, les problématiques rencontrées ainsi que l'explication de vos codes.

### Description détaillée du projet :

#### I – Récupérer les informations du serveur UDP

Un serveur UDP en broadcast (192.168.40.120:40100) concentre toutes les commandes de pizzas des clients sous le format « *Date Heure,ID\_Client,Nom\_Pizza,Taille,Quantité,Heure\_Livraison\_Souhaitée* ».

Exemple de commande envoyée par le serveur UDP :

26/11/2025 10:03:12,530080,Orientale,G,4,10:40

Votre premier objectif est de récupérer les informations disponibles sur le serveur en rédigeant le code Python d'un client UDP. Vous aurez alors en temps-réel les commandes de pizzas de près de 300 clients.

Pour votre programme (*ClientUDP.py*), vous pourrez utiliser la fonction *split()* pour récupérer vos variables *Date*, *Heure*, *ID\_Client*, *Nom\_Pizza*, *Taille*, *Quantité* et *Heure\_Livraison\_Souhaitée*.

#### II – Premiers pas avec une Base de Données

Prenez en main le langage SQL en étudiant le programme *CodePython\_BDD\_PEI.py* (disponible dans le Moodle du Projet). L'étude des fonctions de ce programme rentrera dans l'évaluation de votre projet.

#### III – Dialoguer avec la Base de Données

Rédigez un second programme (*ClientSQL.py*) qui vous permette d'accéder à la table *Client* de la BDD *UE\_ENS\_PROJET* (192.168.40.117:5432) et de lire le temps de livraison (champ *Distance* – en minute) des clients. La table *Client* est sous le format :

Champs :	ID	Distance
Type :	INT	INT
Valeur (ex.) :	529996	24
Valeur (ex.) :	530111	12

Tab. Structure de la table Client avec un exemple d'entrées

Récupérez par la suite le temps de production des pizzas commandées par les clients (champ *TPsProd* – en minute) en accédant à la table *Pizza*. Cette table est sous le format :

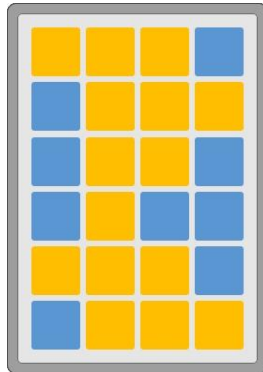
Champs :	Nom	Taille	Composition	TPsProd	Prix
Type :	TEXT	TEXT	TEXT	INT	INT
Valeur (ex.) :	4_Fromages	G	"JJJB,BJJJ,BJJB,BJBB,JJJB,BJJJ"	9	12
Valeur (ex.) :	Chevre	M	"----,-JJ-,JJVJ,BVJJ,-BB-,---"	7	8

Tab. Structure de la table Pizza avec un exemple d'entrées

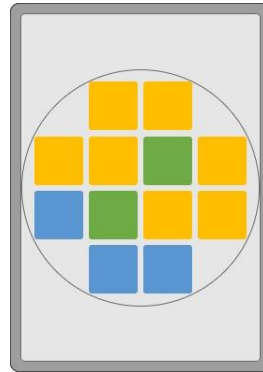
La table *Pizza* comporte des informations sur ; le prix des pizzas (en euro), leurs compositions et leurs temps de production (en minute). Le tout en fonction des tailles des pizzas ; *G* pour « grande » et *M* pour « moyenne ».

Pour les deux pizzas présentées dans le tableau précédent (*4\_Fromages* et *Chevre*), leur composition peut s'illustrer selon la représentation ci-après. Pour cela, le champ *Composition* représente *en ligne* les ingrédients utilisés dans les pizzas ;

- 'J' pour *jaune* peut correspondre à de l'emmental ou à des tranches d'ananas
- 'V' pour des poivrons, des olives, etc.
- Etc.



*Représentation de la composition de la pizza 4\_Fromages (taille G)*



*Représentation de la composition de la pizza Chevre (taille M)*

*Fig. Représentation de pizzas selon leur champ Composition (Table Pizza)*

La finalité du programme *ClientSQL.py* sera de créer une fonction mixant vos deux requêtes SQL. Ainsi vous validerez, pour l'heure de livraison souhaitée par le client, la faisabilité d'une livraison ; selon le temps de production de la pizza (en prenant en compte sa taille) et du temps de livraison au client.

*Faisabilité = Votre\_fonction(ID\_Client, Nom\_Pizza, Quantite, Taille\_Pizza, Heure\_Livraison\_Souhaitée)*

#### IV – Concaténation des programmes « Clients »

Concaténez vos deux programmes clients (UDP et SQL) pour commencer à réaliser le programme du projet qui ; lorsque le serveur UDP envoie une nouvelle commande, vous puissiez définir si celle-ci est "*acceptée*" ou "*non-acceptée*". Réalisez alors une IHM textuelle simple sur votre Terminal Linux.

#### V – Capacités de production et choix du poste de production

L'usine qui produit les pizzas possède six postes de production. Des informations sur ces postes sont stockées dans la table *Production*.

Sur ces postes, il est possible de fabriquer plusieurs pizzas en même temps, c'est ce que l'on nomme « Capacité ». Un poste peut fabriquer une ou plusieurs tailles de pizza (Champ *Taille* = *M* ou *G*, ou les deux : « - »). Pour causes d'allergies, certains postes ne peuvent pas produire certaines pizzas ; le champ *Restriction* indique alors les pizzas non-fabricables par le poste (« - » s'il n'y a pas de restriction particulière). Enfin, le champ *Disponibilite* informe si le poste est opérationnel (=TRUE) ou non.

Champs :	Poste	Capacite	Disponibilite	Taille	Restriction
Type :	INT	TEXT	BOOL	TEXT	TEXT
Valeur (ex) :	3	18	TRUE	G	Chevre,4_Fromages
Valeur (ex) :	4	15	FALSE	-	-

*Tab. Structure de la table Production avec un exemple d'entrées*

Créez des fonctions vous permettant d'attribuer les commandes clients aux bons postes de production. Vous prendrez en compte les restrictions des postes ainsi que leur capacité maximale et leur disponibilité.

Aidez-vous des *listes* sous Python afin d'allouer les commandes de pizzas aux postes ;

Exemple d'utilisation des fonctions *append()* et *remove()* de la classe *list* ;

```
My_list = []  
My_list.append(['Abc',12])  
My_list.append([17,'D','E'])  
My_list.remove(['Abc',12])  
print(My_list)  
print(My_list[0][1])
```

*Note : Retrouvez en Annexe 1 une aide plus complète sur les fonctions de la classe list.*

Ainsi, par poste, utilisez des *listes* pour stocker (au minima) ; la quantité ainsi que l'heure de fin de production (*heure de la commande + temps de production de la pizza*) de chaque commande attribuée au poste.

Supprimez la commande de votre liste lorsque l'heure de fin de production est passée. Utilisez la bibliothèque *datetime* de Python pour gérer des événements sur des horaires.

Mettez à jour votre fonction *Faisabilité* afin de tester si au moins un poste peut prendre la commande client.

## VI – Statistiques et IHM (*Interface Humain-Machine*)

En adaptant le code *ExServeurTCP\_HTML.py* disponible sur le Moodle, développez en *html* un *dashboard* simple, affichant le nombre total de commandes acceptées et refusées. Pour ce faire, vous devrez faire communiquer vos deux programmes (programme *ClientsUDP\_SQL* et le programme de l'IHM). Au choix ; vous pouvez faire passer leurs informations par *UDP* ou par un fichier *JSON*.

*Note : Retrouvez en Annexe 2 une présentation succincte de la structure d'un document HTML.*

## Codes Optionnels

Pour ce projet, vous devrez réaliser au minimum l'une des options décrites ci-dessous. Vous avez le choix de réaliser celle qui vous semble la plus pertinente selon les compétences que vous souhaitez développer.

### **(Option 1) – Statistiques avancées & Amélioration de l'IHM**

Réalisez des statistiques sur la gestion des commandes (Nombre d'ingrédients utilisés par poste, Total des ventes, Quantité produite, etc.) et intégrez ces informations à votre IHM.

### **(Option 2) – Gestion de listes d'attente**

Améliorez la gestion des commandes lorsque la capacité des postes est dépassée, en mettant en attente les commandes qui pourraient être réalisables plus tard ; c.à.d. lorsque la capacité des postes le permettrait.

### **(Option 3) – Optimiser la commande des clients**

Certains clients commandent plusieurs pizzas. Celles-ci peuvent être produites sur différents postes à différents moments. Assurez-vous que le timing de production permette au client de recevoir toutes ses pizzas bien chaudes.

## ANNEXES

---

### Annexe 1

### python List Method Visualised

[●●●●].append(●)	→	[●●●●●]
[●●●●].insert(2,●)	→	[●●●●●]
[●●●●].remove(●)	→	[●●●]
[●●●●].pop()	→	[●●●]
[●●●●].pop(1)	→	[●●●]
[●●●●].reverse()	→	[●●●●]
[●●].extend([●●])	→	[●●●●]
len([●●●●])	→	4
[●●●●][1]	→	●
[●●●●][-1]	→	●
[●●●●][1:3]	→	[●●]
[●●●●][2:]	→	[●●]
[●●●●][:3]	→	[●●●]
[●●]*3	→	[●●●●●●]
list(set([●●●●]))	→	[●●]
[●●●●].index(●)	→	2
[●●●●].count(●)	→	3
[●●●●].clear()	→	[]

## Annexe 2

# Structure HTML ce qu'il faut savoir

