**Style globale choisi :**

Dans le cadre de ce projet et en tenant compte des besoins du client et les exigences de qualité mentionnées au début du rapport nous avons opté pour le style de client serveur comme style global

Un client a besoin de communiquer avec le cluster Kubernetes qui est considéré comme serveur pour notre cas :le binôme a besoin d’accéder à l’application qui sera un service à l'intérieur du cluster (serveur).

* Tout d’abord le client accède à l’application web qui est hébergée à l'intérieur du cluster (mais reste indépendante du reste des applications cibles tournant sur le même cluster) et sera accessible par le réseau LAN de l’école
* Pour l’authentification le binôme fait entrer ses identifiants au niveau de l’interface de l’application web et le serveur de cette dernière confirme l’authentification en accédant à la base de donnée où sont stockées les informations de l’utilisateur et allume son pod en changeant le nombre de réplicats à 1 lorsqu'on est l’utilisateur est connecté , 0 si l’utilisateur est déconnecté
* Le serveur web envoie une réponse au client pour exécuter l’application du vnc viewer et accéder au service de l’application cible exemple packet tracer, via un serveur vnc.

**la justification:**

l’architecture du client serveur se base principalement sur un client qui envoie des requêtes et le serveur qui les exécute, il est adapté pour les applications qui communiquent à travers le réseau .

les avantages du style client serveur est choisi car:

* **une meilleure sécurité** : les points d'accès au cluster sont via une adresse ip unique juste en changeant l’intervalle de port pour chaque application au moment de déploiement selon la nécessité du module
* **une administration au niveau serveur** : le cluster kubernetes a besoin d'être administrer le déploiement et la gestion d'accès au poids et au cluster .
* **un réseau qui peut être étendu:** on peut supprimer ou ajouter autant qu’on veut de binômes sans beaucoup de changement au niveau de performance

le diagramme ci-dessous montre comment ce style est vu d’une manière globale :

**style d’architecture du côté serveur:**

Les applications dans le cluster de serveurs sont vues comme des microservices . Chaque application associée à un module est déployée comme étant un service à part entière .

pour chaque service on crée des pods tels qu’un pod est associé à un seul binôme (ou monôme)

**Pour justifier ce style:**

* chaque instance de l'application est considérée comme un service executable d'une manière autonome et indépendante des autres application
* Les services ont des codes indépendants des autres services chaque service est créé à partir d'une image docker spécifique
* Les services sont responsables de gérer leurs propres données, dans ce cas là chaque service accède à son volume pour persistance de données comme par exemple déposer le fichier de tp dans son espace.
* Bien que les services pour ce cas de projets ne communiquent pas entre eux mais il n'y a une possibilité de communication si on considère que la communication entre packet tracer et wireshark se fait à travers des pods différents chacun déployé d'une manière indépendante du reste des services
* Et un autre point important pour considérer l'architecture des microservice est que chaque service utilise des technologies et dépendances différentes
* De plus, pour plus d'adaptation aux micro services, kubernetes par défaut un orchestrateur est utilisé pour la gestion et l’équilibrage de charge au niveau des nœuds du cluster.