TerriSTORY

Documentation TerriSTORY®

docs.terristory.fr

Architecture logicielle

Architecture logicielle

Présentation de l'architecture logicielle de l'application

L'architecture d'un logiciel décrit la manière dont sont agencés les différents éléments d'une application et comment ils interagissent entre eux. Elle décrit d'une manière symbolique et schématique l'interrelation et l'interaction des différents éléments de système, les modèles et les techniques utilisées pour concevoir et créer une application.

L'application TerriSTORY®

TerriSTORY® est une application à page unique (**SPA**) fondée sur une architecture **3 tiers.**

Qu'est-ce qu'une architecture 3 tiers ?

Une architecture client-serveur représente l'environnement dans lequel des applications de machines clientes communiquent avec des applications de machines de type serveurs.

Si certaines ressources sont présentes sur un deuxième serveur, on parle d'architecture 3 tiers.

L'utilisateur interroge le premier serveur web qui lui-même interroge le deuxième serveur (par exemple un serveur accueillant des bases de données).

Architecture 3 tiers

Qu'est-ce qu'une application à page unique ?

Une application à page unique est différente des applications multi-pages (MPA). Ces dernières sont des applications Web avec plusieurs pages rechargées lorsqu'un utilisateur demande de nouvelles données. Chaque clic de l'utilisateur déclenche donc une requête HTTP vers le serveur. Le résultat de cette nouvelle requête est un rafraîchissement complet de la page, même si une partie du contenu reste inchangée.

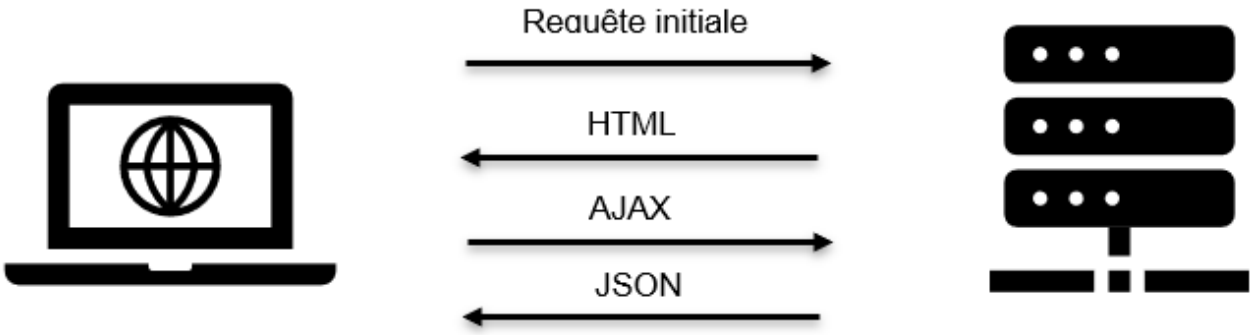
Une application à page unique est une application qui interagit avec les utilisateurs en réécrivant les pages Web existantes avec de nouvelles données provenant du serveur Web chargées et ajoutées aux pages en réponse aux actions de l'utilisateur, au lieu d'utiliser la technique par défaut du navigateur qui exécute une toute nouvelle page.

De cette façon, les utilisateurs peuvent afficher un site Web sans charger l'intégralité de la nouvelle page et des données du serveur. En conséquence, les performances en sont améliorées ce qui offre aux utilisateurs une expérience Web plus dynamique et des transitions *a priori* plus rapides entre les contenus consultés.

Comment fonctionne les SPA ?

Supposons que vous souhaitiez visiter une page Web spécifique. Lorsque vous entrez son adresse, le navigateur envoie cette requête à un serveur qui vous livre un document HTML en retour.

À l'aide d'une SPA, le serveur envoie le document HTML uniquement pour la première requête, et pour les requêtes suivantes, il envoie des données JSON. Cela signifie qu'une SPA réécrira le contenu de la page actuelle et ne rechargera pas toute la page Web. Par conséquent, pas besoin d'attendre beaucoup de temps pour le rechargement.



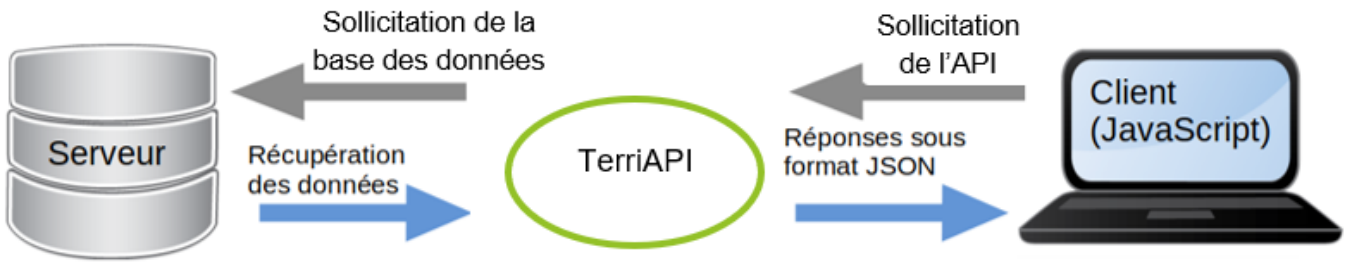
Architecture SAP

Technologies employées

La partie serveur de l'application **TerriSTORY®** est développée en **Python.** Son rôle est de solliciter les bases de données et d'en extraire toutes les informations souhaitées, puis de les transmettre à la deuxième partie dite « partie client » développée avec la librairie **React** en **JavaScript** qui les met en forme et les restitue sous formes graphiques et cartographiques à l'utilisateur. Le langage de la partie serveur est interprété par le serveur au sein duquel le site web est hébergé tandis que la partie client est interprétée par le navigateur internet de l'utilisateur.

Interactions entre React et les scripts chargés de solliciter les bases de données

Lorsqu'un utilisateur recourt à l'un des différents modules de TerriSTORY®, il interroge au moyen d'une requête HTTP (c'est-à-dire construite à partir d'une adresse URL), lancée par le JavaScript, une partie de l'API, c'est-à-dire un ensemble de fonctions chargées de transmettre les données que nous souhaitons mettre en forme à la partie client au format JSON. Ces fonctions écrites en Python ainsi lancées permettent d'encapsuler le reste de l'application en faisant appel à d'autres fonctions d'un autre module que nous nommons « *controller* ». Notons que c'est à l'intérieur de ce module auquel l'API fait appel que sont écrites les requêtes SQL.



Interactions client-serveur

Il est d'ailleurs possible de représenter les interactions entre ces différents objets et leur structuration dans le temps au moyen d'un diagramme de séquences.

Les traits pleins représentent les tâches asynchrones confiées à un élément de l'architecture logicielle. Ces tâches ont pour but de faire appel à une autre brique logicielle et de provoquer une réponse sous forme d'échanges de données. Ces réponses sont représentées dans le diagramme sous forme de traits pointillées.

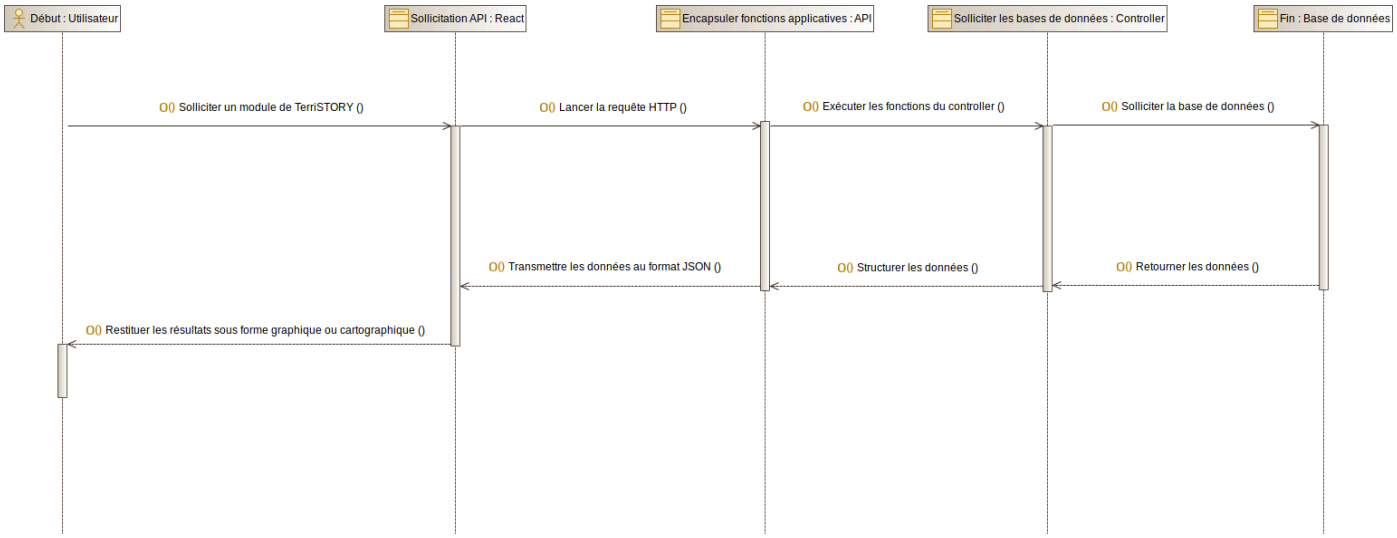


Diagramme de séquence

Développer une application de manière asynchrone présente l'avantage de pouvoir exécuter des tâches en parallèles. Les interactions entre les parties client et serveur d'un site web sont toujours asynchrones.

Dépendances logicielles

Framework front-end

ReactJS est la bibliothèque utilisée pour le *front-end* : c'est une librairie javascript libre qui gère l'interface de l'application, elle facilite la création d'application web monopage, via la création de composants dépendant d'un état et générant une page HTML à chaque changement d'état.

Nous recourrons également à **nodejs**, une plate-forme logicielle libre qui permet d'exécuter du JavaScript côté serveur. Nous nous servons notamment du gestionnaire de paquets associé qui permet d'automatiser le téléchargement des dépendances ainsi que leur mise à jour.

En outre, la compilation du code de l'application est réalisée avec **Webpack**, un logiciel qui fonctionne dans l'environnement nodejs.

Framework back-end

Sanic est le framework Web Python utilisée pour gérer des réponses HTTP rapides via une gestion asynchrone des requêtes.

Système de gestion de base de données

PostgreSQL est le système de gestion de bases de données (SGBD) mobilisé pour gérer les bases de données qui contiennent les fonds de plan OSM et les données nécessaires au fonctionnement des différents modules de TerriSTORY®. Elles sont sollicitées au moyen de requêtes SQL exécutées au sein de fonctions Python. Nous utilisons également l'extension **PostGIS** pour pouvoir utiliser des données géographiques.

Les échanges de données entre ces deux grandes parties sont assurés au format JSON par un module Python de Sanic (Alembic).

Déploiement de l'application

L'application est déployée sur trois serveurs hébergés chez OVH

- Le serveur de développement qui héberge une version de TerriSTORY mise à jour à chaque modification du code de la branche principale (master).
- Le serveur de test, qui joue le rôle de serveur de pré production.
- Le serveur de production qui héberge la version publique.

Nous disposons d'un script **ansible** de déploiement automatique nommé « **gitlat-ci.yml** »

La configuration automatique des trois serveurs est assurée par un script **ansible** qui contient des paramètres cryptés que seules les personnes qui disposent des clés correctes peuvent lire.

Docker

Nous recourrons à Docker pour afficher les tuiles OSM et géométries des territoires via l'application **Postile** développée par Oslandia.