Rapport D’architecture
Projet BRAIN

FILOCHE Léo
MORLOT-PINTA Louis
DE ZORDO Benjamin
LEGRAND Quentin
PINAULT guillaume

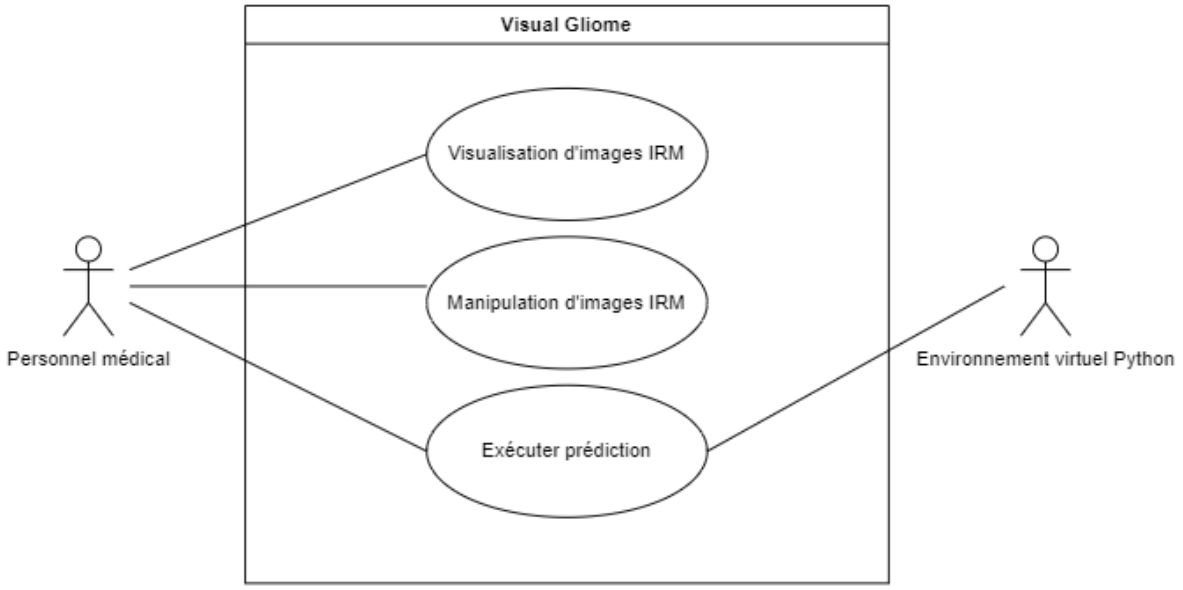
Ce rapport à pour objectif de décrire le design de de la solution et son architecture.

I. Présentation du design de la solution

a. Objectifs de la solution

La solution doit répondre à deux objectifs distincts. Le premier est de désigner un framework de deep learning performant, capable de segmenter d'éventuels gliomes a partir d'images volumétriques IRM.

Le second est de proposer une application mobile permettant de visualiser et de manipuler les images IRM, ainsi que de communiquer avec le framework pour obtenir les prédictions.



b. Architecture de la solution

L'architecture de la solution se découpe donc en deux parties. D'un côté on retrouve une pipeline de framework de réseau de neurones, ce qui se traduit par un ensemble de scripts python regroupant les post et pré traitement, les scripts d'exécution et de formatage de données. Cette pipeline sera hébergée sur un environnement virtuel python sur une machine possédant les capacités de calcul adéquates à la segmentation.

Pour la partie application mobile nous avons fait le choix de créer un site web qui permettrait au personnel soignant, une fois connecté, d'accéder au radios IRM du patient et d'exécuter une prédiction à partir des ces radios et d'un framework de deep learning. Ce site web serait composé du front et d'un back end, ainsi que d'une base de données. On a aussi fait le choix d'un cache pour héberger les radios IRM visualisées.

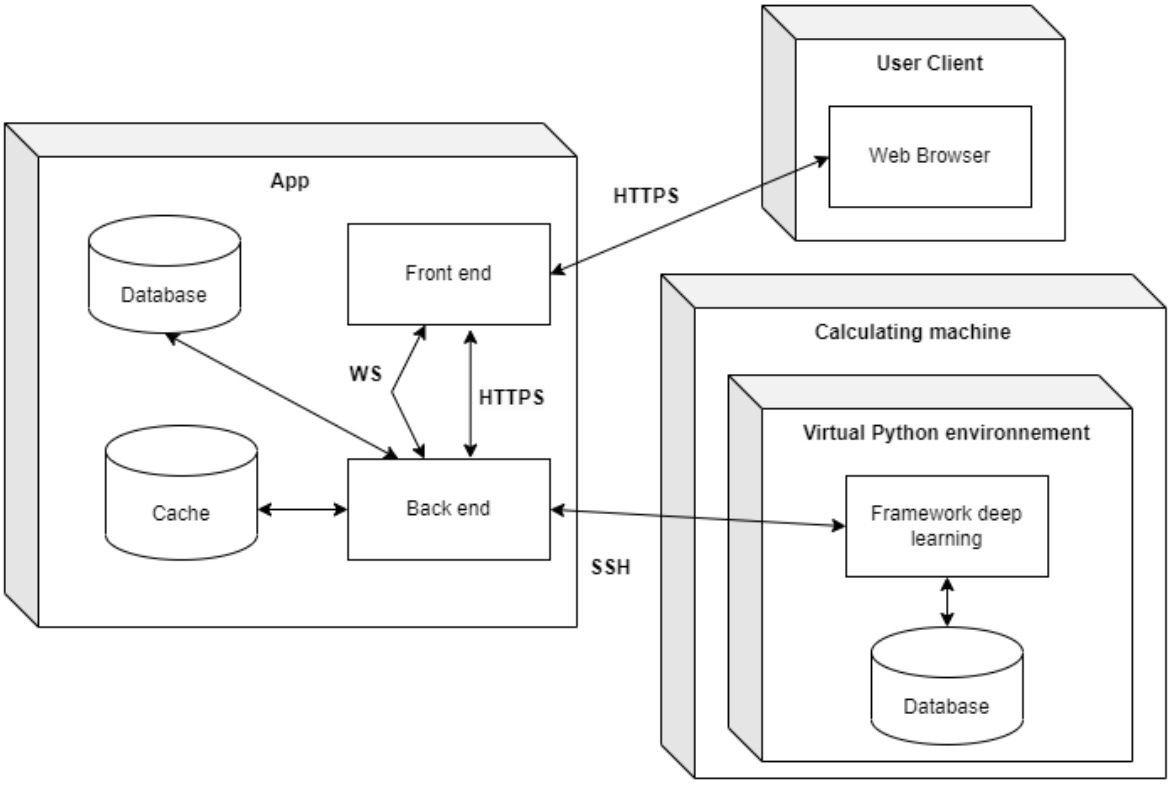


Diagramme de déploiement

Au vu de l'architecture, voici le déroulement d'un cas d'utilisation. On suppose que le personnel médical est déjà connecté et que ses identifiants ont été confrontés à la base de données des utilisateurs. Le Personnel sélectionne alors l'ensemble de radios IRM à traiter. Le site web envoie alors une requête avec les données IRM. Cette requête est transmise au framework de deep learning. Sur le chemin, les données sont stockées dans un cache pour ensuite être transmises au visualiseur d'IRM.

Une fois que la prédiction est effectuée par le framework, on remonte la prédiction jusqu'au front end pour afficher le résultat et que l'utilisateur puisse comparer avec les IRM chargées dans le visualiseur.

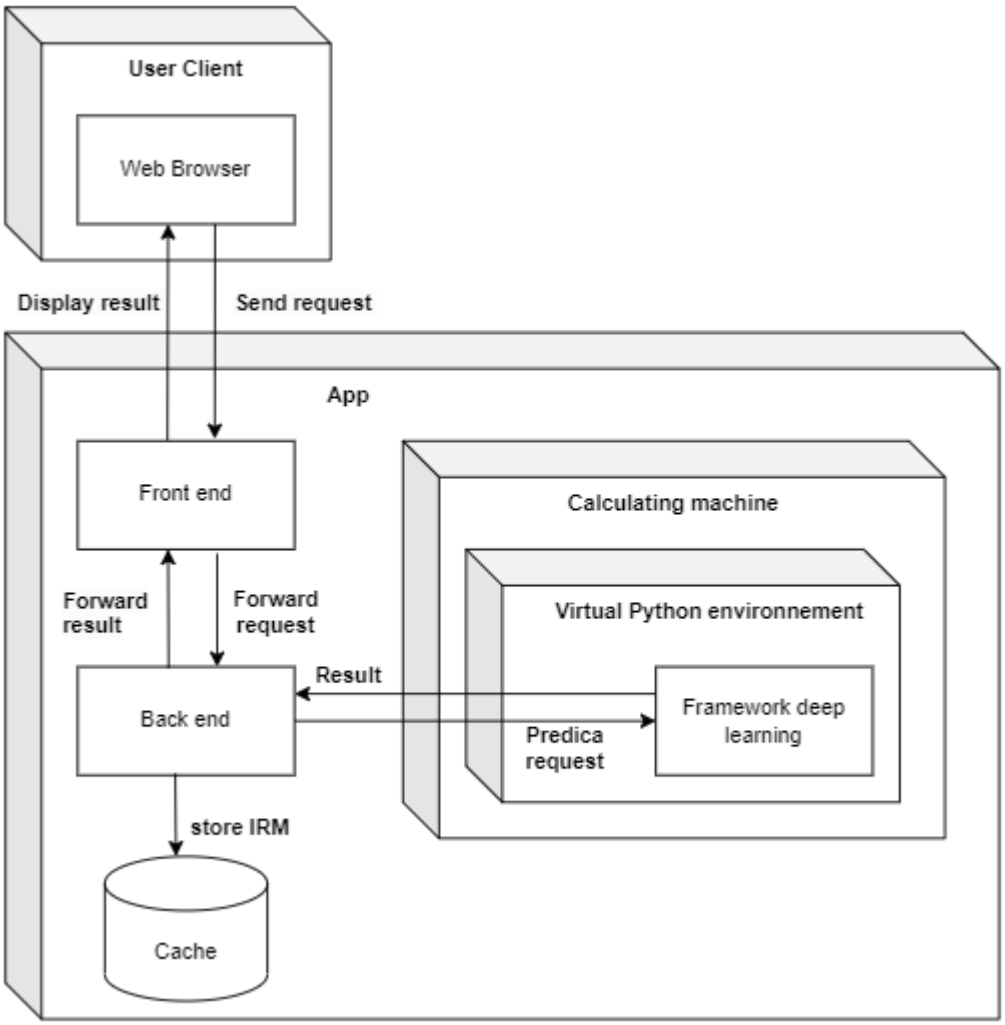


diagramme de composant

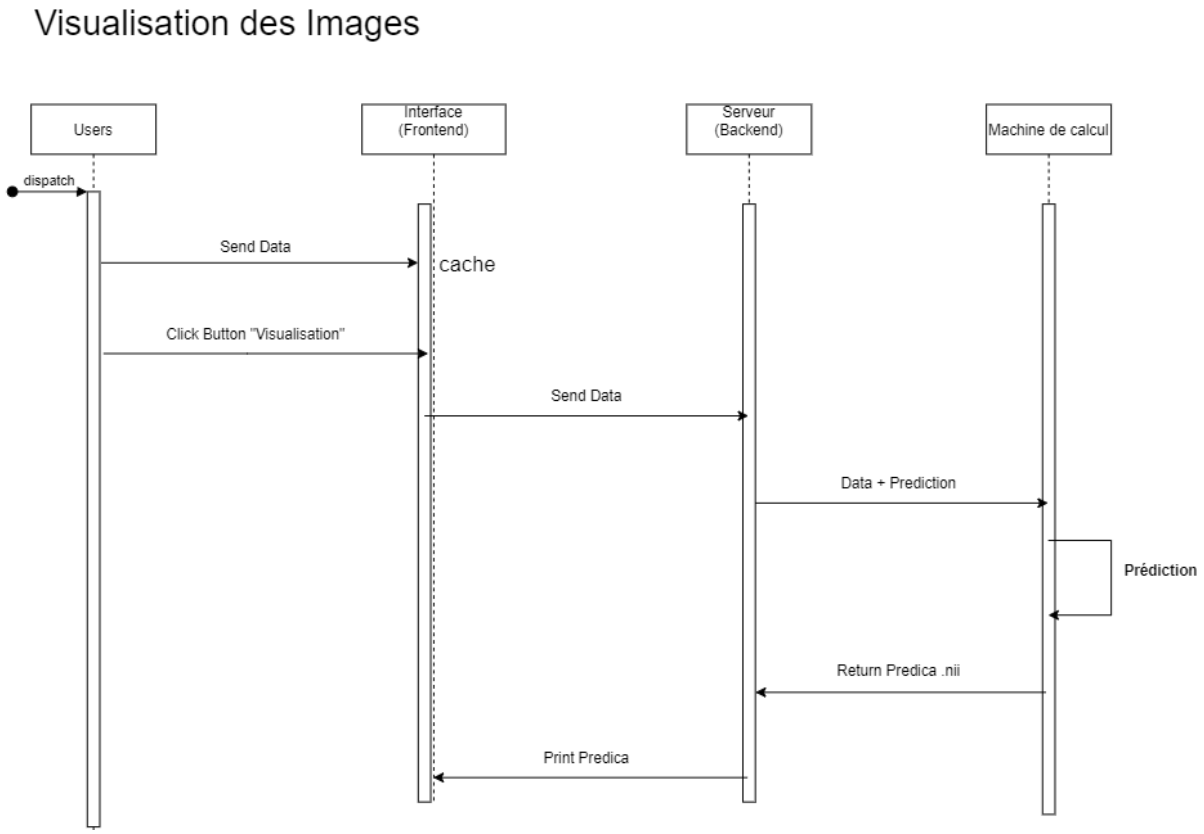


diagramme de séquence

c. Choix techniques

Au niveau des choix techniques, pour ce qui est de la partie Framework deep learning. Nous avons décidé de travailler en python et d'implémenter un modèle de réseau de neurones convolutif appelé U-NET. Nous avons choisi ce modèle suite à l'étude NVIDIA qui a comparé différents modèles et traitements pour obtenir les meilleurs résultats en termes de segmentation d'imagerie médicale. La pipeline et les données d'entraînement sera host sur la machine de calcul de l'istic.

Pour la partie app mobile, nous avons porté nos choix sur des technos classiques telles que Nest pour le backend, angular pour le front end, SqLite pour la base de données et Redis pour le cache. La partie visualiseur de fichier nii, format des images médicales, sera gérée par une bibliothèque JS appelé NIFTI-reader.

II. Mise en place de la solution

a. Conception des sprints

Nous avons rapidement séparé l'équipe en deux. Une première équipe LEGRAND Quentin et DE ZORDO Benjamin est chargée du développement du framework de deep learning et la seconde équipe, composée du reste des membres du groupe, est chargée de l'app mobile, d'établir la connection avec l'environnement virtuel host sur la machine de calcul et d'implémenter un visualiseur de fichier nii.

A Chaque début de séance les deux équipes procèdent à une courte réunion pour rendre compte de leurs avancées et des éventuels problèmes rencontrés. Nous nous sommes accordés pour concevoir les sprints suivants.

- **Sprint 0 :** Maîtrise du domaine métier et des notions D'IA. Mise en place de l'environnement de travail
- **Sprint 1 :** Entraîner le modèle et effectuer une première prédiction. Prototype d'interface Web et de visualiseur.
- **Sprint 2 :** Portage de la pipeline du framework sur la machine de l'istic. Mise en place de la communication SSH entre la machine de l'istic et l'interface Web
- **Sprint 3 :** Ajustement des pré et post traitements pour améliorer la précision du framework. Amélioration de l'ergonomie et de l'esthétique de l'interface web et du visualiseur.
- **Sprint 4 :** Tous les modules sont déployables et opérationnels

b. Cadre de production

Nous avons basé notre cadre de production sur l'outil de versionnage et d'organisation GitHub. L'interface web sera développée par les outils, IDE, souhaités par les membres de l'équipe.

La configuration du projet s'articule autour des données d'input fournies par le client de l'utilisateur. Le framework renvoie alors une prédiction qui sera affichée sur le front end et manipulable dans le visualiseur.

Pour la livraison, toute la pipeline et les données d'entraînement du modèle seront stockées sur la machine de l'istic. La partie de l'interface web sera elle livrée sous la forme d'un docker composé. Cela assure la fiabilité d'exécution du logiciel à la livraison.

c. Contraintes de production

Nous sommes cependant confrontés à de nombreuses contraintes. Concernant la partie framework on retrouve des contraintes de performance, un framework non fiable est inacceptable quand la vie de patients est en jeu. Cependant, après un état de l'art il en est sorti que le meilleur modèle est celui optimisé pour des GPU NVIDIA car désigné par ces derniers. On a donc une contrainte matérielle et une contrainte d'adaptation.

Pour la partie interface web la seule contraintes repose sur le visualiseur qui ne présente que peu de solutions. Nous avons décidé d'adapter une bibliothèque JS pour éviter d'avoir à traduire des bibliothèques en C via Web Assembly.