La Nécropole Numérique - Compte-rendu d'activité - 2021-2022

Christelle Seng, Rachid El-Hajaoui, Caroline Font, Thomas Guillemard, Cyrille Le Forestier, Mercey Florent

Recherche et développement

R133050

Inrap
Institut national de recherches archéologiques préventives

La Nécropole Numérique

Christelle Seng, Rachid El-Hajaoui, Caroline Font, Thomas Guillemard, Cyrille Le Forestier, Florent Mercey

Avec la collaboration de Micheline Kerien et Frédéric Barenghi

Compte-rendu d'activité
2021 - 2022

Sommaire

1. Résumé des épisodes précédents

Le présent compte rendu ici fait suite à celui réalisé en 2020[1], des difficultés organisationnelles ne nous ayant pas permis la rédaction d'un compte rendu en 2021. Ainsi, l'ensemble des évolutions et modifications réalisées dans le courant 2021 et 2022 sera abordé ici.

1.1 La base de données, état des lieux à la fin 2020

L'année 2020 avait permis la réalisation d'une première version de la base de données adaptée aux travaux anthropologiques, tant sur le terrain qu'en laboratoire. Pour mémoire, l'architecture retenue est une base de données au format Spatialite, interfacée dans Qgis, dont les tables peuvent être divisées en trois ensembles :

– Les couches minimales utilisées à l'Inrap (dites "6 Couches"), qui comportent des données géométriques et des données attributaires essentielles (soit un numéro d'identifiant, un type et/ou une nature)

– Un ensemble de 25 tables nommé Badass (Base Archéologique de Données Attributaires et Spatiales) permettant l'enregistrement des données archéologiques attributaires (fait, us, sondages, mobilier, etc.) et de récupérer la géométrie présente dans les 6 couches,

– Un ensemble de 8 tables (dites Of The Dead - OTD) regroupant les données funéraires.

L'utilisation d'un format spatialite conditionne l'emploi d'un certain nombre de processus automatisés. Les bases de données spatialite/sqlite et Postgresql/postgis utilisent des *triggers* (ou déclencheurs), qui sont des scripts directement implémentés dans la base au format SQL et qui automatisent certaines procédures. La création, la suppression ou la modification d'une valeur dans un champ peut ainsi se répliquer dans d'autres champs, d'autres tables sous l'action d'un *trigger*. L'usage de ces *triggers* est l'un des fondements du fonctionnement de Badass et de OTD et assure l'intégrité des données saisies.

A l'issue de l'année 2020, une première version de la base de données était opérationnelle. La réflexion concernant les tables du module OTD n'était cependant pas tout à fait aboutie et leur interaction avec les tables de BADASS demandait à être testée. Cependant, un des principes fondamentaux, c'est à dire l'enregistrement à l'échelle de l'os, était retenu[2].

Une première ébauche de formulaire sous Qgis avait été réalisée mais elle ne répondait pas tout à fait aux attentes en termes d'interaction homme-machine. Un projet de création de formulaire sous QT/python avait donc été élaboré avec de premières maquettes de formulaires. Cependant, aucun des membres de l'équipe ne disposait des connaissances techniques pour en assurer le développement.

Concernant l'acquisition spatiale, les protocoles liés à la photogrammétrie (prise de vue et traitements) ont été affinés et les options retenues confirmées.

Le projet s'est poursuivi en 2021 et 2022, approfondissant et corrigeant les premiers développements ici résumés.

1 Christelle Seng, Rachid El-Hajaoui, Caroline Font, Thomas Guillemard, Cyrille Le Forestier, et al.. La Nécropole Numérique - Compte-rendu d'activité - 2020. [Interne] Inrap. 2020. ⊛hal-03620076⊛

2 Ce principe a été employé par Rozenn Colleter, Jean-Baptiste Romain, Jean-Baptiste Barreau (INRAP, CNRS, UMR 6566, Colleter et al 2020) dans le cadre du projet HumanOs

2. Structuration des données attributaires et spatiales

2.1 Année 2021

Si le projet s'est articulé en 2021 autour de plusieurs objectifs, il a également été l'occasion d'éprouver sur le terrain le fonctionnement de la base de données en testant les interfaces, débuguant plusieurs fonctionnalités et en en développant de nouvelles.

Il a été possible de réaliser l'enregistrement des données taphonomiques et anthropologiques, ainsi que de vérifier le fonctionnement du cœur de l'application, et tester les interfaces des tables Badass en condition opérationnelle.

2.1.1 La nécropole des Mastraits

Des moyens ont été engagés en début d'année afin de finaliser une version fonctionnelle de la partie anthropologique (OTD) de la base de données SpatiaLite. Les modifications concernant la partie anthropologique se traduisent par l'ajout ou la suppression de tables, de champs, mais aussi par le développement de triggers spécifiques à la partie funéraire.

Une première version est testée au cours de la campagne de fouille programmée de la rue des Mastraits de Noisy-le-Grand en juin 2021. Cette première version est modifiée et corrigée au cours de l'été 2021 puis consolidée en octobre. L'idée d'une saisie à l'aide d'un mannequin est proposée, et une première version est implémentée dans la base de données. La saisie sur mannequin n'est cependant pas opérationnelle à la fin de 2021.

Les tables et la structure de la base de données en octobre 2021 sont illustrées ci-dessous (**Tab. 1**, **Fig. 1**).

Table	Objectif
man_squel	dédiée à l'enregistrement des données du mannequin de saisie, le mannequin représente le squelette dans son entier
man_part_ana	recense les parties anatomiques du mannequin
man_os	les ossements du mannequin. Chaque ossement est dessiné dans Qgis et peut donc être sélectionné individuellement. Le code SQL permettant de générer les tables de la base de données intègre ces informations spatiales ce qui permet, dès la génération d'un nouveau projet, de disposer d'une couche comportant le dessin de chaque ossement.
t_us_sep	fait le lien entre le numéro d'US et les éléments contenus dans la sépulture. Au sein d'une sépulture, un squelette est considéré comme une US à part entière. On établit un lien de 1 à 1 entre la table t_us et t_us_sep
t_squelette	enregistre les informations générales du squelette
t_obs_sep_prim	regroupe les observations dans le cas d'une sépulture primaire
t_obs_sep_sec	regroupe les informations en cas de sépulture secondaire
nombre_sec	dénombre les individus dans une sépulture secondaire
t_part_anatomiq	crée une relation entre un squelette et une partie anatomique (table man_part_ana)
t_os	les informations à l'échelle de l'os
t_mob_sep	les informations concernant le mobilier de la sépulture
t_amenag_sep	les informations concernant les aménagements de la sépulture
t_cont_sep	les informations sur le contenant
t_mesure	enregistrement des différentes mesures à l'échelle de l'os
t_patho	observations concernant les pathologies, les caractères discrets, les marqueurs d'activité ou de stress, à l'échelle du squelette, de l'os ou de la partie anatomique
t_sexe	les critères et observations permettant d'identifier le sexe de l'individu
t_age	les informations permettant de déterminer l'âge au décès
otd_thesaurus	destinée à rassembler l'ensemble des termes employés dans la base de données afin de générer des listes de valeur.

Tab. 1 Liste des tables associées à l'extension OTD en 2021

Ces données sont reliées à t_squelette par le biais du numéro de squelette, qui correspond au numéro d'US.

Concernant l'interface de saisie, les formulaires élaborés fin 2020 sont affinés au tout début 2021. Deux étudiants en informatique à l'Université de Rennes 1 (IUT de Saint-Malo et Istic), encadrés par Jean-Baptiste Barreau (CNRS), rejoignent le projet dans le cadre d'un stage et commencent à développer un plugin destiné d'une part à faciliter le déploiement de la base de donnée dans QGIS, et d'autre part à mettre en place des formulaires de saisie. Ce plugin fait appel au langage Python, utilisé par Qgis.

Malgré une importante avancée quant au fonctionnement du plugin et le travail fourni par les stagiaires, le plugin n'est pas fonctionnel à la fin des deux stages, fin mai 2021 (renvoi aux rapports de stage.) : la réflexion autour de cet outil a en effet évolué au cours des deux stages, ne permettant pas aux programmeurs d'achever l'outil dans le temps leur étant imparti.

Afin de proposer un outil opérationnel pour la campagne de fouille 2021, des formulaires temporaires sont donc rapidement réalisés grâce aux outils natifs de qgis. Des sous formulaires permettant la saisie des données taphonomiques (dans les tables t_squelette et t_obs_sep_prim) sont intégrés au formulaire d'enregistrement des US (t_us) (**Fig. 2**).

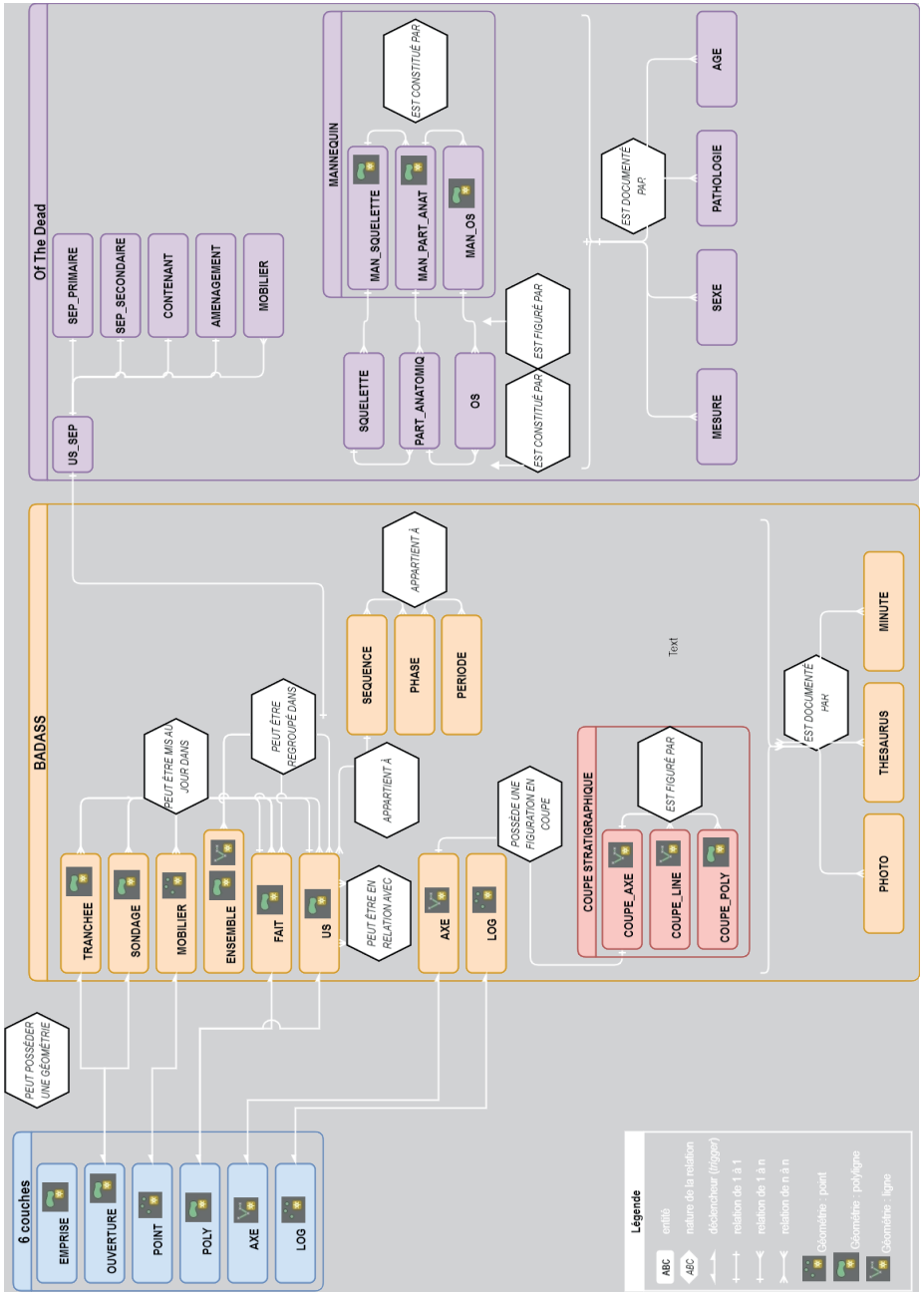


Fig. 1 Modèle conceptuel simplifié de la base de données en 2021

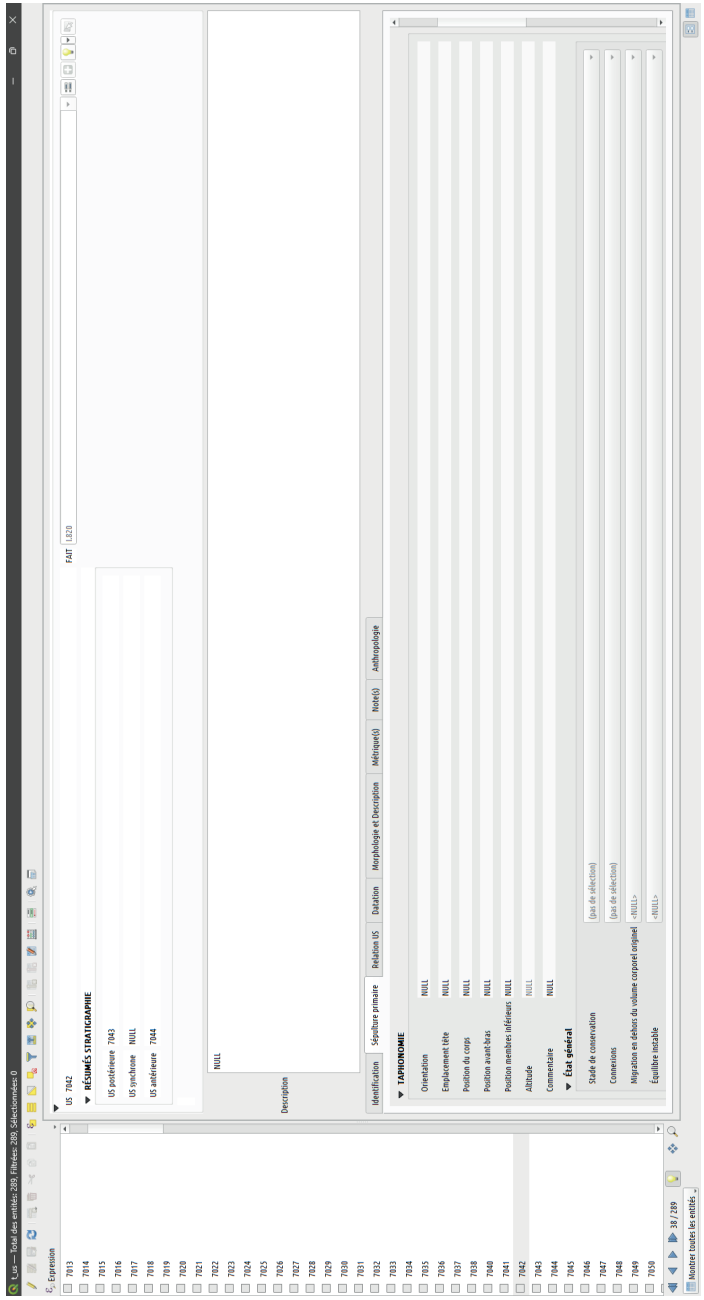


Fig. 2 Le formulaire de saisie de la table US pour la campagne de fouille de 2021

Il a ainsi été possible de tester la saisie de données directement sur le terrain par deux opérateurs possédant une bonne connaissance de Qgis, en l'absence d'interface définitive. Un débogage réactif et continu de la part des concepteurs a permis de corriger les quelques dysfonctionnements rencontrés pendant la fouille, et a mis en évidence quelques désordres structurels dans la base de données. Ces constatations ont donné lieu à la suppression de certains triggers dans un premier temps, puis à une réévaluation du fonctionnement de la base de données par la suite.

La saisie des données de terrain a été partiellement complétée dans le cadre du rapport intermédiaire de la fouille programmée.

En août 2021, une première expérimentation de saisie des données anthropologiques et biologiques d'un individu a été effectuée en laboratoire (métrique, détermination du sexe, etc.). Là encore, il est apparu que des ajustements devaient être effectués, notamment au niveau des filtres conditionnant la saisie des mesures ostéologiques.

La restructuration du module d'enregistrement funéraire s'inscrivait donc au programme de 2022.

Cette première expérience a cependant permis d'envisager de tester la base de données en condition d'opération préventive, sur des opérations de petite envergure.

2.1.2 Des tests en contexte préventif

Les expérimentations menées en contexte préventif n'avaient pas pour but d'intégrer les données funéraires, l'extension OTD étant instable. Il s'agissait surtout d'éprouver la structure de Badass et l'utilisation des triggers avec d'autres types de données et d'adapter les formulaires de saisie.

- une saisie expérimentale des données de diagnostic à Meaux (une tranchée en contexte urbain) (RO : C. Delozanne) en post-fouille et vectorisation des coupes ;
- une expérimentation de saisie numérique directe dans le cadre d'une fouille préventive urbaine à Meaux (RO: C. Seng, RS: C. Le Forestier). Bien que le contexte du site soit propice à l'expérimentation du module funéraire (nécropole), il n'a pas été mis en œuvre sur le terrain. Il sera utilisé en post-fouille, une fois la restructuration effectuée ;
- une expérimentation de la base de données pour une opération archéologique sur le bâti avec des formulaires et des thésaurus adaptés sur la fouille de Ferrières-en-Gâtinais dans le Loiret (R.O. : Carole Lallet).

2.1.3 Bilan de 2021

L'expérimentation de l'année 2021 a permis un premier déploiement de la base de données et de proposer des outils de saisie plus ergonomiques que lors de la campagne 2019. Les différents tests sur le terrain et lors de travaux de laboratoire, ont permis d'en apprécier les bénéfices ainsi que les limites: la stabilité et la robustesse du cœur de la base de données, qui s'est avérée parfaitement fonctionnelle mais aussi la nécessité de repenser le mode de saisie des données funéraires et donc de remodeler une partie des tables dédiées. Les éléments recueillis ont ainsi permis de préciser le programme de travail pour l'année 2022.

Les moyens ont été utilisés tout au long de l'année (ajustements du code avant la phase de terrain, expérimentation pendant la campagne de fouille, les autres tâches ne répondant pas aux exigences d'un calendrier précis…). La possibilité de travailler à nouveau en présentiel a grandement facilité les différents échanges, aussi bien en interne qu'avec nos collaborateurs externes.

2.2 Année 2022

2.2.1 Des tests en contexte préventif

D'autres tests en contexte préventif ont été menés en 2022 avec une saisie directement en phase de terrain[3].

- Sur un diagnostic à Orléans (R.O. : Mathilde Noël)
- Sur un diagnostic à Meaux (R.O. : Christelle Seng)
- Sur un diagnostic à Sully-sur-Loire (R.O. : Florent Mercey)
- Sur un diagnostic à Hanches (R.O. : Laurent Fournier)

Ces différentes expérimentations ont été précieuses pour l'amélioration de l'ergonomie de saisie et l'ajout de nouvelles requêtes de Badass (hors extension funéraire).

Par ailleurs, sur une opération, la saisie s'est effectuée en post-fouille : diagnostic à Meaux (R.O. : Christel Delozanne).

3 Un grand merci aux collègues volontaires et courageux qui se sont prêtés aux expérimentations.

2.2.2 Structuration de la base de données

A la suite de ces expérimentations, une session de travail a été organisée sur une semaine en avril 2022 afin de corriger les bugs identifiés, de forme comme de fond.

Il en résulte une très importante reprise de la structure des tables de l'extension OTD et des *triggers* afférents, ainsi que quelques modifications structurelles nécessaires dans le code du noyau principal de la base de données.

2.2.2.1 Modifications générales dans Badass

Les modifications effectuées sur la base de données principale sont peu nombreuses et répondent à de petits dysfonctionnements (ajustement de triggers) ou à une volonté de simplification (renommage de certains champs).

La photogrammétrie constituant la base des relevés scientifiques dans le cadre de l'expérimentation, l'ajout d'une table destinée au suivi et à l'inventaire des modèles photogrammétriques est apparue nécessaire. La mise en relation entre cette table d'inventaire et les tables d'enregistrement des photos et des minutes de terrain est à réaliser. Cette table, t_photogram est décrite ci-dessous (**Tab. 2**).

Champ	Type	Commentaire
id_photogram	entier	clé primaire
numodel	texte	nom ou numero du modele
datemodel	date	date de constitution du modèle
legend	texte	légende affectée au modèle
obj_model	texte	sujet du modèle
job	texte	nom du fichier
urljob	texte	emplacement du fichier
creator	texte	auteur du modèle
comment	texte	commentaire
boitier	texte	référence de l'appareil photo
iso	nombre	sensibilité iso utilisée
nettete	texte	distance de mise au point
objectif	texte	longueur focale de l'objectif
diaphragme	texte	ouverture du diaphragme
temps_pose	texte	vitesse d'ouverture
xmlcalib	texte	chemin vers le fichier de calibration
convers	entier	état de conversion de format
suivi	texte	étape de traitement du modèle

Tab. 2 Liste des champs pour la table t_photogram

Des vues récapitulatives ont également été ajoutées afin de synthétiser au fur et à mesure les données saisies dans les différentes tables (**Fig. 3**). Elles permettent de générer des inventaires ou d'aider au suivi de l'enregistrement (relations stratigraphiques, erreurs de saisies de relations stratigraphiques, faits par tranchée, minutes, photos, us par faits).

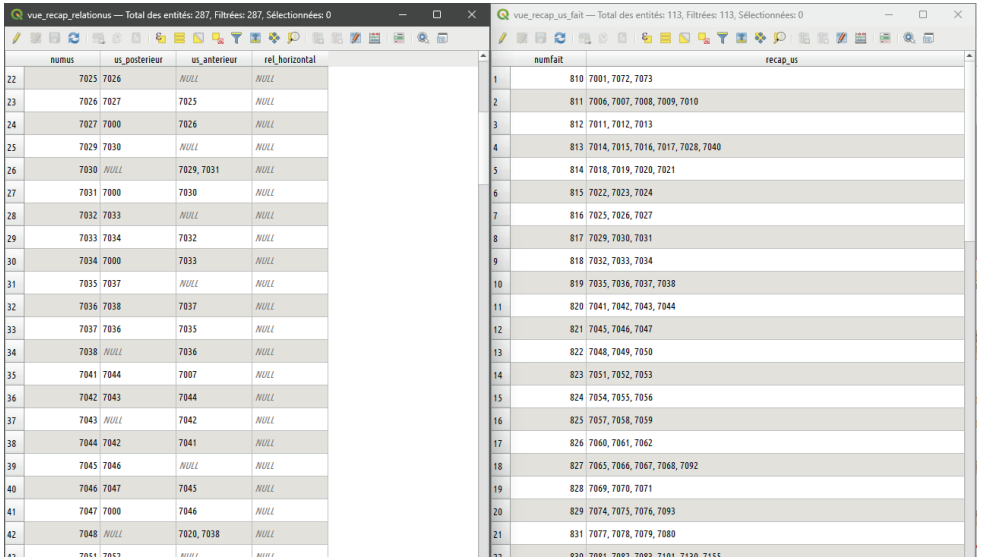


Fig. 3 Exemple de requêtes récapitulatives enregistrées dans la base de données

2.2.2.2 Restructuration de l'extension funéraire

Plusieurs tables ont été supprimées, de nouvelles ont été créées et de nombreux champs renommés.

Le nombre de tables de OTD a été réduit à douze et la structure simplifiée (**Tab. 3**, **Fig. 4**).

Table	Objectif
mannequin	modèle pour extraction des données vers la table t_os
t_squelette	enregistre les informations générales du squelette.
t_obs_sep	regroupe les observations sur les sépultures
t_os	regroupe les informations à l'échelle de l'os
t_cont_sep	regroupe les informations sur le contenant
t_mesure	enregistrement des différentes mesures à l'échelle de l'os
t_patho	observations concernant les pathologies, à l'échelle du squelette, de l'os ou de la partie anatomique
t_sexe	les critères et observations permettant d'identifier le sexe de l'individu
t_age	les informations permettant de déterminer l'âge au décès
otd_thesaurus	destiné à rassembler l'ensemble des termes employés dans la base de données afin de générer des listes de valeur
model_anthropo	liste des enregistrements à créer par défaut dans les tables t_obs_sep ou t_contenant

Tab. 3 Liste des tables associées à l'extension OTD en 2021

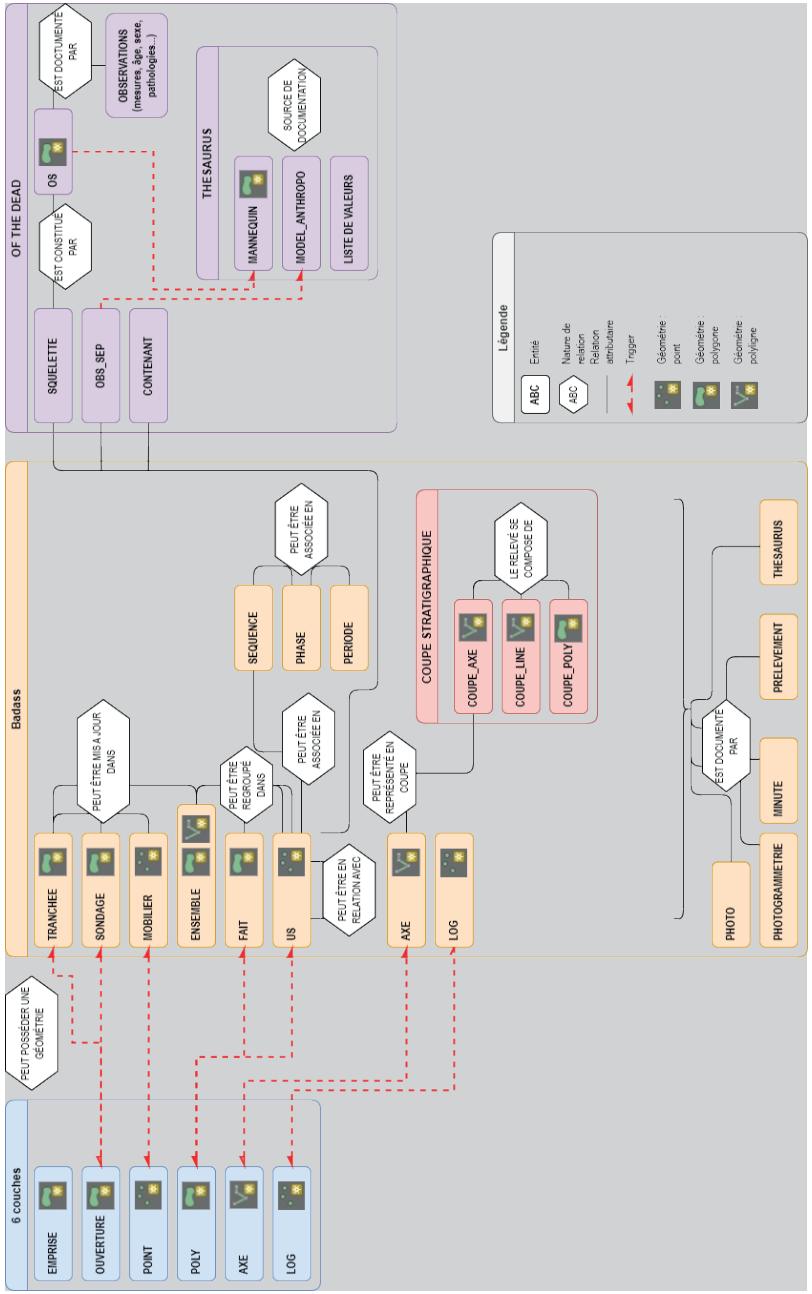


Fig. 4 Modèle conceptuel simplifié de la base de données en 2021

Les tables man_part_ana, man_os et man_squel ont été fusionnées en une seule table mannequin, qui comprend l'ensemble des ossements du corps humain, identifiés notamment selon l'âge des individus (périnatal, immature, adulte.). Il s'agit d'une table de référence destinée à alimenter la table t_os, à la manière d'un thesaurus géométrique (**Fig. 5**).

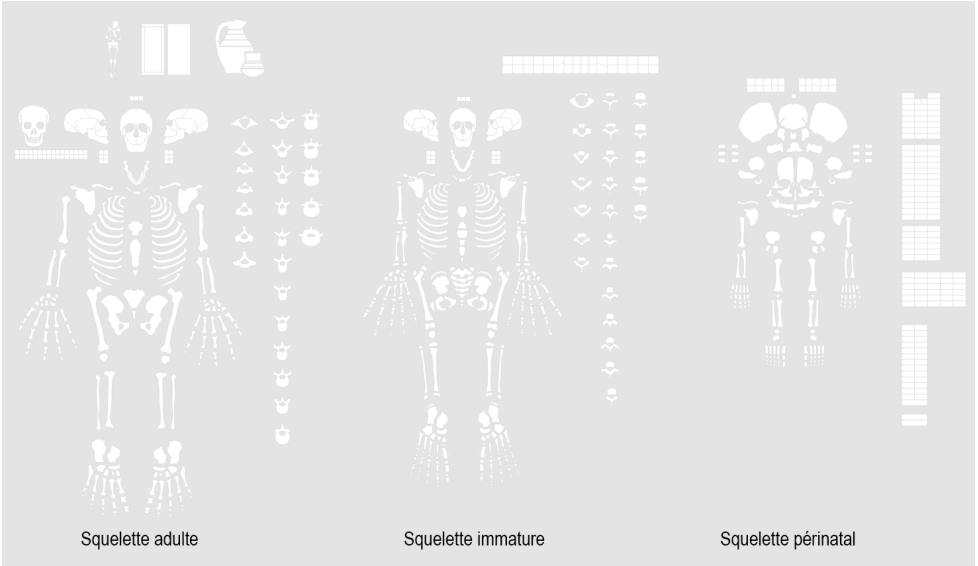


Fig. 5 Les mannequins de démontage anthropologique intégrées à la base de données

La table t_us_sep a été supprimée. Le lien entre l'US et le dépôt funéraire est réalisé directement entre la table t_us et la table t_squelette via le champ sq_numus de la table t_squelette. Un trigger permet d'alimenter directement le numéro du dépôt funéraire par le numéro de l'US dès lors qu'on identifie l'US comme un dépôt funéraire, qu'il soit primaire ou secondaire[4].

La table t_squelette a vu l'adjonction de nombreux champs, notamment pour des mesures ou différents indices réalisés à l'échelle du squelette. Ainsi, l'identification de certains caractères sexuels ou liés à l'âge ont été regroupés dans cette table. Les tables t_pathos, t_sexe et t_age, qui recueillaient ces observations, ont été supprimées, tout comme la table t_part_anatomiq, qui regroupait les observations à l'échelle des parties anatomiques (finalement inutiles).

Les deux tables t_obs_sep_prim et t_obs_sep_sec ont été fusionnées en une seule, t_obs_sep. Elle ne contient que peu de champs, listés ci-dessous (**Tab. 4**).

Champ	Type	Commentaire
id_obs_sep	entier	clé primaire
id_model	entier	clé étrangère : renvoie vers la table model_anthropo qui contient la liste des observations qu'il est possible de réaliser.
sq_numus	entier	clé étrangère : numéro du squelette, c'est à dire le numéro d'US
sep_nature	texte	il s'agit de renseigner ici le type d'observation effectuée. Cette valeur est issue de la table model_anthropo
sep_result_text	texte	il s'agit de renseigner les résultats de l'observation, ici sous forme de texte
sep_result_num	entier	des résultats numériques de l'observation
sep_comment	texte	commentaire spécifique au caractère du dépôt du squelette

Tab. 4 Liste des champs associés à la table t_obs_sep

Cette table est destinée à enregistrer les observations pouvant être réalisées sur une sépulture ou sur son contenant. Chaque enregistrement correspond donc à un type d'observation, qui est renseigné dans le champ sep_nature. Le principe de cette table évite ainsi la création de très nombreux champs dans la table t_squelette, champs qui ne sont pas tous renseignés puisque certains ne s'appliquent que dans le cas de sépulture primaire, ou uniquement en présence d'un contenant, etc. Ainsi, plutôt que de créer plusieurs tables et de très nombreux champs qui, du fait de la multitude de situations rencontrées, ne seront pas tous renseignés, nous créons plusieurs observations dont la nature varie en fonction du sujet et du cadre d'observation. La cardinalité entre l'unité observée (la sépulture, le contenant) et la nature de l'observation est ainsi un lien classique de base de données, de 1 à n (**Fig. 6**).

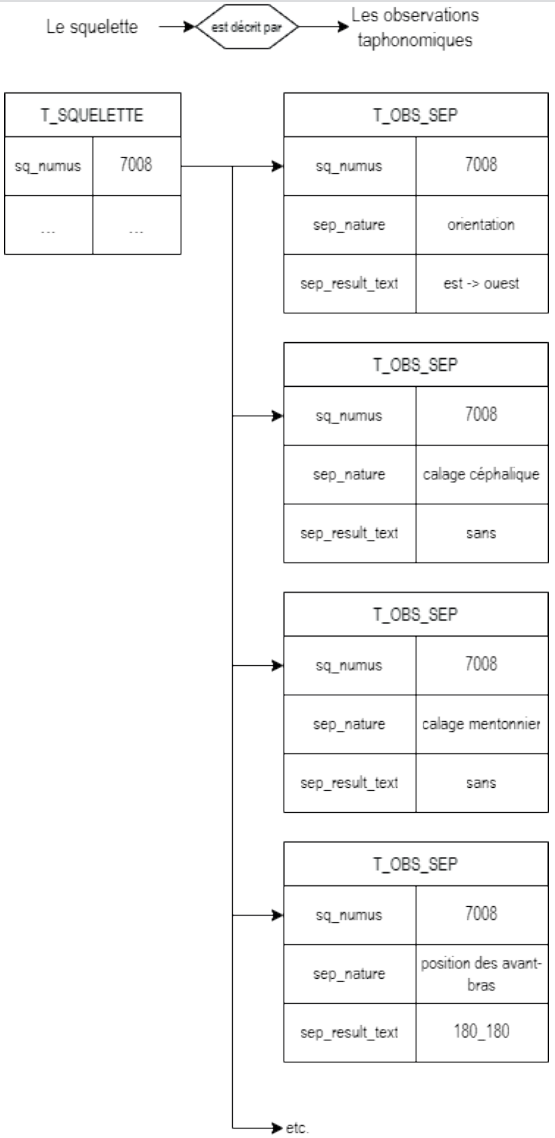


Fig. 6 Principe d'enregistrement des observations faites sur le dépôt

4 Nous enregistrons bien ici le geste funéraire, le fait de déposer l'individu dans la fosse sépulcrale, ou de déplacer les ossements en position secondaire.

Partant du constat que les observations diffèrent en fonction de la nature de la sépulture, afin de guider l'utilisateur dans sa saisie, il nous est apparu intéressant de proposer la génération automatique des observations attendues selon la nature de la sépulture[5].

La table model_anthropo, contient la liste des différentes observations pouvant être réalisées sur une sépulture, en fonction de sa nature, primaire ou secondaire par exemple.

La structure de la table model_anthropo est décrite ci-dessous (**Tab. 5**).

Champ	Type	Commentaire
id_model	entier	clé primaire
enregistrement	texte	nom de l'enregistrement à créer, nature de l'observation
ordre	entier	permettant de rajouter un numéro d'ordre
primaire	entier	permet de distinguer les enregistrement pour les sépultures primaires
secondaire	entier	permet de distinguer les enregistrement pour les sépultures secondaire
contenant	entier	permet de distinguer les enregistrements pour les contenants
commentaire	texte	commentaire associé à l'observation (définition, préconisation)
affich	entier	permet d'afficher ou non l'entité dans un formulaire (pour Qgis)

Tab. 5 Liste des champs associés à la table t_obs_sep

C'est un trigger qui, en fonction de la nature de la sépulture enregistrée dans la table t_us (primaire, secondaire…), génère automatiquement la liste des observations dans la table t_obs_sep depuis la table model_anthropo (**Fig. 7**).

Cette structuration donne une grande souplesse aux utilisateurs, qui peuvent décider de créer leurs propres observations et de modifier la liste des observations au sein de la table model_anthropo. Le module funéraire peut donc être adapté à chaque contexte.

La table t_os a été simplifiée. Toutes les observations, qui avaient été intégrées à cette table ont été réparties entre les tables t_obs_sep et t_squelette. La table t_os joue cependant un rôle important dans la génération des fiches de conservation, un des buts de l'extension.

En effet, chaque fois que l'âge d'un individu est renseigné dans la table t_squelette, (périnatal, immature ou adulte), un trigger peuple la table t_os par l'ensemble des ossements issus de la table mannequin. Le numéro d'US de squelette est automatiquement attribué aux os créés. Chaque enregistrement de la table t_os correspond donc à un os spécifique d'un squelette donné[6]. Le fait qu'il dispose d'une géométrie permet d'interagir spatialement avec cet os, de le modifier, le couper, ou de le supprimer.

La table t_mesure est destinée à recueillir l'ensemble des informations de mesure à l'échelle de l'os, depuis la table t_os (cf *infra*).

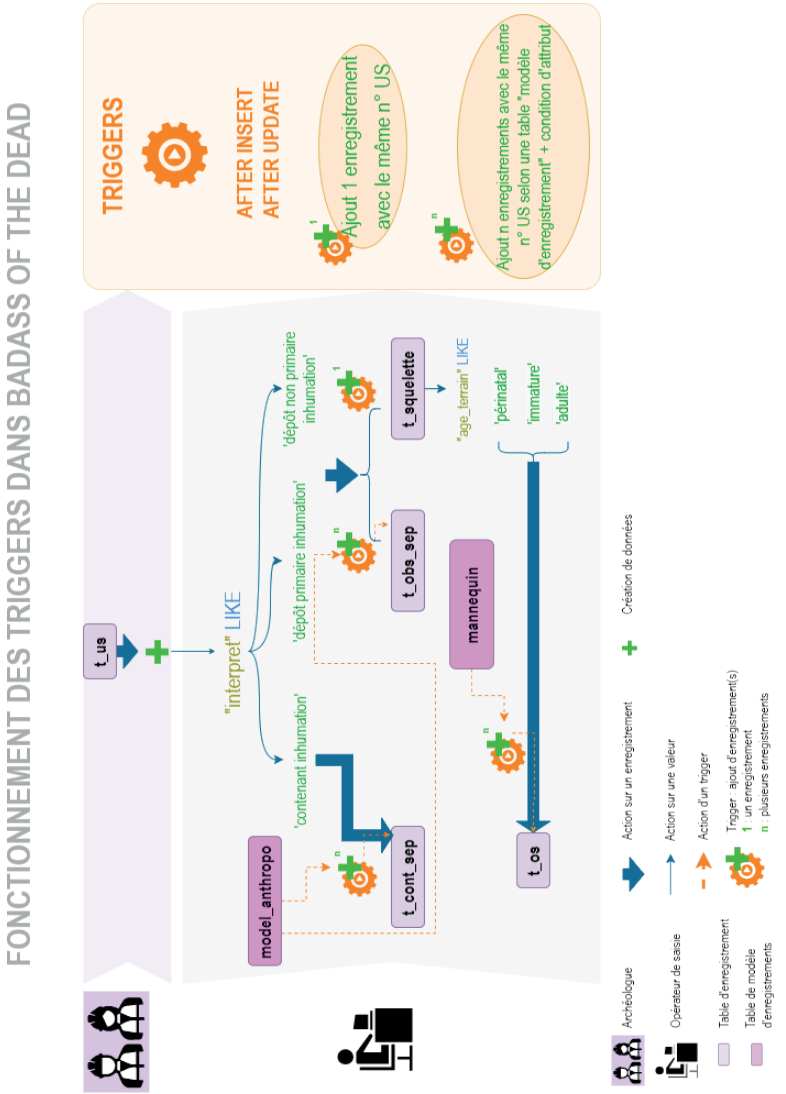


Fig. 7 Principe de fonctionnement de l'ensemble des triggers dans OTD

5 Les expérimentations de Badass sur le terrain, en contexte préventif, ont permis de développer ce principe.
6 Ceci signifie que l'intégration des 766 sépultures recensées sur Noisy implique pour cette table la création de 188 400 enregistrements environ.…

2.2.3 Mannequin de saisie, t_os et t_mesure

A l'image des fiches anthropologiques utilisées pour le démontage ou l'enregistrement d'informations en laboratoire, il nous a semblé nécessaire de mettre en place un système permettant la saisie de ces informations dans Qgis. Après quelques réflexions, la méthode suivante a été conçue.

Une première couche, mannequin, déjà complète dans la base de données, recueille la représentation spatiale des ossements pour des individus adultes, immatures, ou périnataux. Trois mannequins sont ainsi représentés. Chaque os est dessiné et dispose de plusieurs caractéristiques présentent dans la table attributaire (**Tab. 6**).

Champ	Type	Commentaire
id_man	entier	clé primaire
catego	texte	la catégorie au sein du mannequin (ex : os, contenant...)
nom	texte	le nom associé à la géométrie (ex : fémur, ulna...)
cote	texte	la latéralisation (ex : droite, gauche)
part_anat	texte	la partie anatomique auquel l'os appartient (ex : membre inférieur, thorax...)
ss_catego	texte	la classe d'âge à laquelle se réfère la géométrie de l'os (ex : adulte, immature, périnatal)

Tab. 6 Liste des champs associés à la table mannequin

Lorsque, dans la couche t_squelette, il est précisé l'âge estimé de l'individu (adulte, immature, périnatal), un *trigger* copie l'ensemble des os du mannequin selon l'âge renseigné dans la couche t_os. Chaque os ainsi créé dans la table t_os comprend donc l'ensemble des informations telles que renseignées dans le mannequin, auquel on associe le numéro de l'individu. Nous obtenons donc, pour chaque individu, un squelette complet, qui peut être modifié (attributaire et spatial).

Les informations contenues dans la table t_os sont présentées ci-dessous (**Tab. 7**).

Champ	Type	Commentaire
id_os	entier	clé primaire
id_man	entier	clé étrangère qui fait référence à l'identifiant de l'os dans le mannequin
sq_numus	entier	numéro d'US du squelette
numindiv	entier	numéro d'individu si nécessaire
nom_part_anat	texte	le nom de la partie anatomique
nom_os	texte	le nom de l'os
orient	texte	une orientation de l'os (s'il n'est pas en position anatomique)
ext_proximal	entier	présence de l'extrémité proximale de l'os
diaphyse	entier	présence de la diaphyse de l'os
ext_distale	entier	présence de l'extrémité distale de l'os
etat_conservation	texte	état de conservation de l'os
raison_absence	texte	raison de l'absence de l'os
comment	texte	commentaire à l'échelle de l'os

Tab. 7 Liste des champs associés à la table t_os

La table t_os contient l'ensemble des ossements de tous les squelettes recensés dans la base avec leur caractéristique, par individu.

L'utilisateur peut dès lors utiliser les données créées dans la table t_os pour renseigner les informations anthropologiques concernant l'individu. Il peut ainsi modifier les ossements, mais également, selon le type d'os, renseigner des observations qui sont intégrées à la table t_mesure (cf *supra*).

Les différentes observations qui peuvent être réalisées ont été classées en plusieurs catégories : stress, pathologie, marqueur d'activité, caractère discret, état sanitaire, mesure. Lors de la création d'une observation sur un os, on choisit la nature de l'observation que l'on souhaite réaliser. Puis, on renseigne le nom de l'observation. Enfin, on enregistre la valeur associée à l'observation (**Fig. 8**).

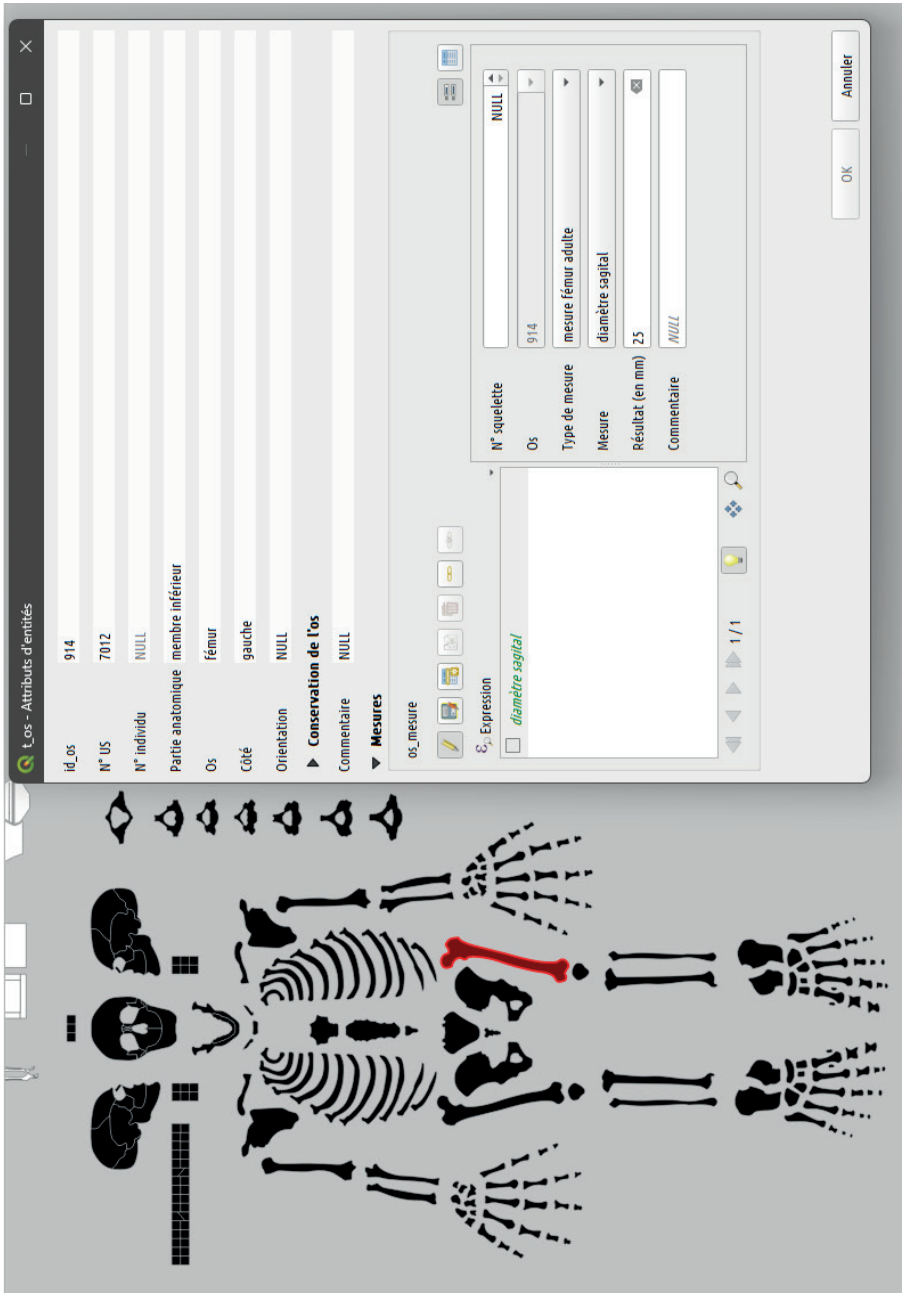


Fig. 8 Exemple d'interface de saisie des observations à l'échelle de l'os

2.2.4 Ergonomie des formulaires

Notre tentative de création de formulaires dans le cadre d'un plugin python avec l'aide de deux étudiants et de Jean-Baptiste Barreau au printemps 2021 n'avait pas pu aboutir. Le travail envisagé s'était avéré considérable, et peu évolutif. En 2021, nous avions donc rapidement réalisé des formulaires avec les outils de Qgis afin que la base de données puisse être exploitée sur le terrain.

Ceci nous a permis de changer de perspective. En effet, les versions de Qgis progressant, il est devenu aujourd'hui plus simple de personnaliser des formulaires pour interagir avec les données. Qui plus est, la gestion des formulaires étant confiée aux styles, l'utilisateur peut créer ses propres formulaires au sein d'un nouveau style, ce qui permet de répondre à des besoins spécifiques et ponctuels tout en conservant les formulaires déjà existants. Plusieurs versions de formulaires peuvent donc être associées à une même couche.

Si la création de formulaire demande une certaine aisance dans Qgis et une bonne connaissance de la structure de la base de données, elle est loin d'être inabordable avec une documentation adaptée, qu'il nous reste à réaliser. Cette solution n'est en outre pas incompatible avec le développement d'interfaces spécifiques[7] qui pourraient être laissées aux bons soins des équipes utilisatrices. En découplant le fond (la base de données) de la forme (l'interface), nous gagnons ainsi en robustesse et en pérennité, là ou des applications telles que Filemaker ou 4D conditionnent l'accès aux données à l'emploi des logiciels utilisés pour le développement de la base.

Nous avons donc, au cours de cette année 2022, repris les formulaires réalisés en 2021 et les avons affinés. Ils ont été conçus dans un souci de simplicité, afin qu'ils puissent facilement s'adapter aux différentes tailles d'écran rencontrés, notamment dans le cas d'usages de tablettes Windows. Ils sont ainsi conçus sur une colonne, à l'image du formulaire d'enregistrement des squelettes.

2.2.4.3 Quelques exemples de formulaire

Face au très grand nombre d'options possibles lors de la réalisation des mesures sur les ossements, et au retour des utilisateurs, la mise en place de filtres ne proposant que certaines mesures s'est imposée (**Fig. 8**).

7 Rappelons que de nombreux langages peuvent être employés pour interagir avec une base de données sqlite/spatialite, de php au javascript, en passant bien sûr par le python par exemple. Mais il est également possible de l'interfacer avec MS access ou libreoffice Base.

2.2.5 L'atlas

Dans le cadre du rapport triennal de la fouille programmée (2019-2022), nous avons utilisé la fonction d'Atlas de Qgis: elle permet, à partir d'une unique mise en page, d'interroger la base de données et de fournir autant de figures que d'enregistrements, générant le catalogue des sépultures fouillées ces trois dernières années.

Chaque page de l'atlas comporte ainsi le dessin de la sépulture, son orthophotographie, une photo générale, les profils, des informations synthétiques concernant la position des bras ou avant bras, ainsi que la description de la sépulture (**Fig. 9**).

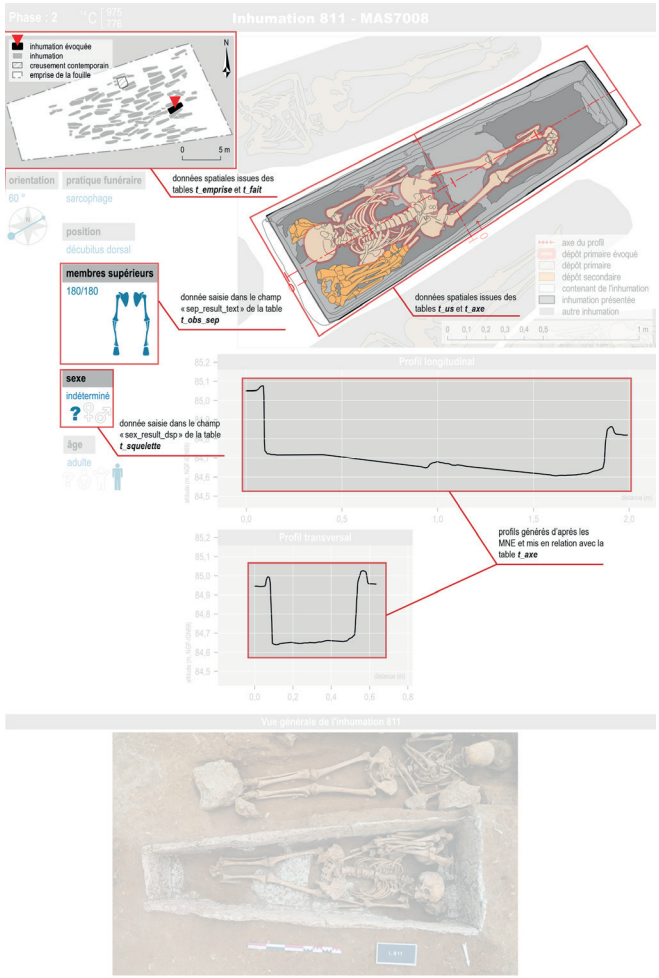


Fig. 9 Exemple de constitution d'un atlas, complété par les différents champs de la base de données

2.2.6 L'extension Badass

La gestion des formulaires par plugin étant abandonnée, nous avons souhaité mettre en place une extension permettant à l'utilisateur de créer simplement un projet qgis comprenant les tables de Badass. Nous avons volontairement limité l'action du plugin aux seules tables Badass, l'extension OTD nécessitant plus de tests avant diffusion. Le projet Qgis et les formulaires eux mêmes ne sont en outre pas totalement finalisés.

L'association Archéologie des Nécropoles, à laquelle est adossée l'organisation de la fouille programmée de Noisy-Le-Grand, a, au cours de l'année 2022, embauché un développeur web, Alex Baiet, pour travailler sur la base de données concernant le handicap[8] mais également tout projet numérique en lien avec la nécropole de Noisy-le-Grand. Nous avons donc sollicité celui-ci pour travailler sur le plugin Qgis.

En s'appuyant sur le travail déjà effectué au printemps 2021 par Emma Chapuis et Alexandre Humeau, Alex Baiet[9] a pu nous proposer, dès la fin du mois de juin 2022, une extension fonctionnelle pour la création de la base de données et du projet associé. Celle-ci est actuellement disponible dans le dépôt des extensions de Qgis (extension expérimentale)[10]. Elle ne fonctionne que sur les versions de Qgis supérieures à 3.28.5.

Cette extension permet, par une fenêtre, de créer ou modifier une base de données (**Fig. 10**).

Fig. 10 Interface de l'extension Badass : accueil

8 https://archeohandi.huma-num.fr/public/accueil
9 https://github.com/alex-baiet/badass/
10 https://plugins.qgis.org/plugins/badass/

En cliquant sur le bouton Créer[11], une nouvelle fenêtre permet de sélectionner l'emplacement du projet ainsi que son nom, et celui de la base de données (**Fig. 11**). L'extension crée la base de données et le projet à l'emplacement spécifié. La fenêtre doit ensuite être fermée.

Cette extension permet donc de disposer rapidement des éléments nécessaires pour travailler avec Badass (**Fig. 12**).

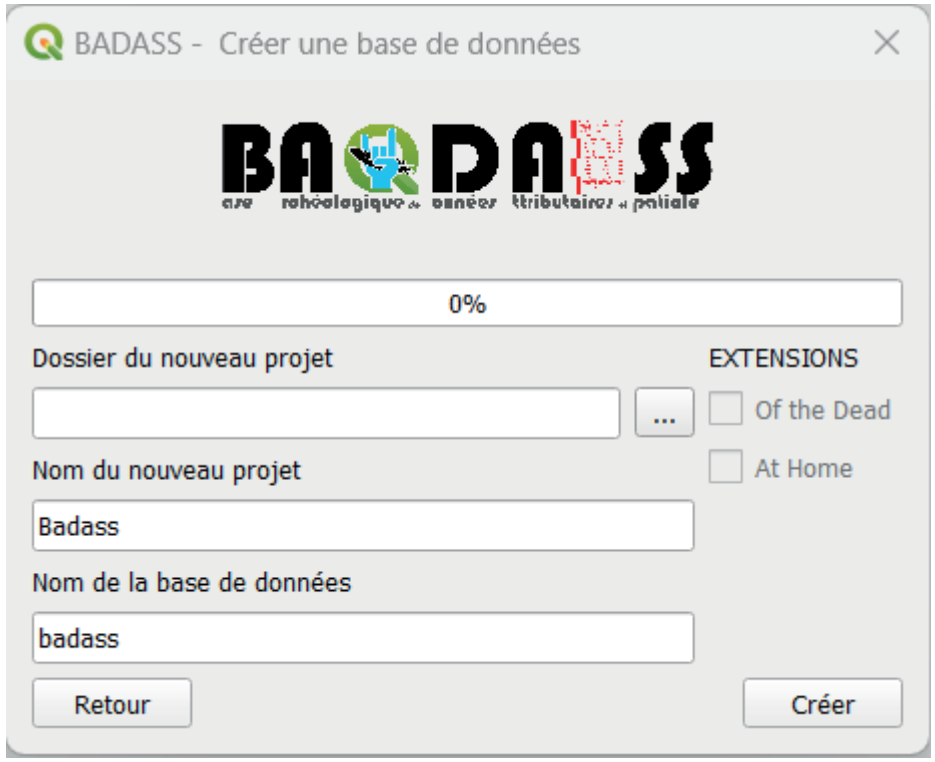


Fig. 11 Interface de création de la base de données et du projet dans l'extension

De futurs développements permettront d'ajouter, lors de la création du projet et de la base de données, les tables permettant la gestion des données anthropologiques (OTD), mais également de modifier la base de données existante si l'on souhaite par l'ajout, à postériori, de ces tables.

Le projet et le code sql permettant de générer la base de données sont modifiables dans le dossier de l'extension, ce qui permet aujourd'hui d'être réactif dès lors qu'un changement survient (changement de version, modification du code, correction de bugs).

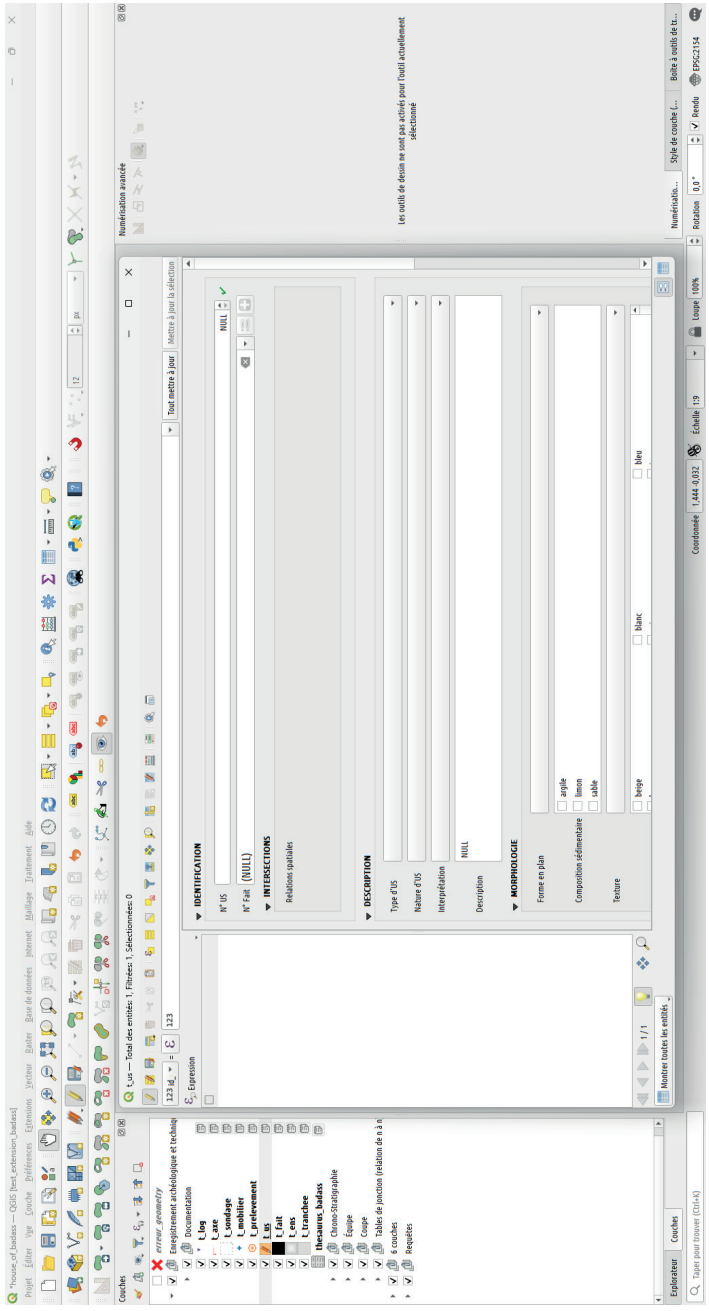


Fig. 12 Projet QGIS créé par l'extension Badass

11 actuellement, le bouton "Modifier" ne fonctionne pas.

2.2.7 Bilan de 2022

Au cours de l'année 2022, nous avons ainsi pu apporter de nombreuses corrections à la base de données. Plusieurs tables ont été supprimées, certaines corrigées, de nouvelles créées. Il en résulte une structure plus simple que celle initialement proposée. Nécessairement, il a fallu corriger plusieurs triggers.

La mise en place du mannequin de saisie a été l'un des points les plus importants de cette année. L'utilisation des triggers a permis la création automatique de l'ensemble des ossements du squelette à l'enregistrement d'une sépulture. Ces ossements peuvent alors recevoir la totalité des informations biologiques et ostéologiques, qu'il s'agisse de mesures, ou d'observations sur les pathologies par exemple.

Un très gros travail a également été réalisé sur l'Atlas, permettant d'automatiser la réalisation du catalogue des sépultures à partir des données contenues dans la base.

Enfin, le développement de l'extension Badass rend celle-ci plus accessible à l'ensemble des utilisateurs de Qgis, tout en ouvrant vers de prochaines évolutions.

3. Levé photogrammétrique, numérisation, génération de la documentation scientifique numérique

3.1 Année 2021

Les protocoles de prises de vue pour la photogrammétrie et les traitements de modélisation ont été affinés. Ils sont efficients, justes et réalisables par les participants à la fouille, qui ont pu expérimenter eux-mêmes l'acquisition des prises de vues pour le levé photogrammétrique et le traitement des modèles, jusqu'à l'édition de la documentation scientifique.

Tous les rasters des sépultures individuelles (MNS et orthoimages) ont été intégrés dans le projet afin de numériser le contour des fosses sur la couche poly : les géométries sont bien répercutées automatiquement sur la table t_fait par le *trigger*.

La numérisation des US (dépôts funéraires, contenant, etc.) et des détails (couche de lignes) sont en cours sur des couches distinctes de la base SpatiaLite, en suivant la méthode de dessin de pierre à pierre proposée par le réseau des référents[12]. Les données spatiales de 2022 seront numérisées selon la méthode utilisée pour la vectorisation des coupes : le modèle graphique qui gère les traitements réduit le nombre de manipulations et semble plus adapté pour récupérer les lignes qui marquent les limites entre les os, après la fusion des polygones[13].

Les protocoles de prises de vue pour la photogrammétrie et les traitements de modélisation sont efficients, justes et réalisables par les archéologues et les bénévoles. Chacun peut ainsi réaliser son propre levé, jusqu'à l'édition des documents scientifiques.

12 https://formationsig.gitlab.io/fiches-techniques/vecteur_dessin/05_pap.html
13 https://formationsig.gitlab.io/fiches-techniques/vecteur_dessin/coupe_SIG/coupes_QGIS.html

3.2 Année 2022

Les protocoles de prise de vues ont été ajustés (points d'amer, paramétrage des prises de vue). Ces dernières ont été organisées de sorte à différencier les séries nécessaires à la réalisation de la documentation scientifique: vues zénithales pour la création de l'orthoimage et vues obliques pour la génération des profils. Des séries complémentaires permettent de compléter le modèle 3D et de créer des documents pour la valorisation.

Sur le terrain, les fouilleurs définissent eux-mêmes le positionnement des points topographiques, paramètrent les réglages de l'appareil photo, définissent le protocole de prises de vues et prennent les photos.

Une collaboration a été initiée avec François Giligny et Ophélie Magnani (Paris1) afin de tester la numérisation des sépultures avec un scanner 3D (**Fig. 13**). Le modèle utilisé est un Artec Leo, un scanner maniable et autonome qui semble adapté aux conditions des chantiers archéologiques. Le temps accordé à la prise de données sur le terrain est équivalent à celui de la photogrammétrie.

Fig. 13 Numérisation d'une sépulture avec un scanner 3D à main par François Giligny (Paris 1)

Les premiers rendus présentent des nuages de points moins complets. Des ajustements sont encore à trouver tant du côté de la collecte de données que du traitement des nuages de points.

Le relevé manuel d'une sépulture a été effectué afin de comparer objectivement les méthodes de relevé (précision du dessin, qualité de l'interprétation, etc.). L'expérience est à réitérer lors des prochaines campagnes : la répétition de ces confrontations permettra d'argumenter en faveur des deux types de relevés.

Enfin, les modèles 3D issus de ces travaux sont mis à jour régulièrement sur la plateforme Sketchfab, en accès libre[14].

14 https://sketchfab.com/ArcheoNec

4. Actions de valorisation et présentations

4.1 L'année 2021

Lors des Journées Européennes du Patrimoine organisées à Noisy-le-Grand les 18 et 19 septembre 2021, les membres franciliens de l'équipe et de nombreux membres d'ADN ont participé aux animations proposées dans le village archéologique. Une exposition virtuelle et interactive mise en place par la mairie de Noisy-le-Grand et accessible sur des bornes présentait notamment une partie des données des campagnes précédentes d'après nos travaux : shapes pour les plans de synthèse et modèles 3D de sépultures sélectionnées (https://experience.arcgis.com/experience/dbec6c7a1e6743998fee12579b61aa64/).

4.2 L'année 2022

– séminaire archéomatique en archéologie funéraire du 2 mars 2022, organisé par le GAAF et les Ateliers Archéomatiques (Tours) (https://www.gaaf-asso.fr/wp-content/uploads/SeminaireGaaf_2mars2022.pdf) ;

– réunion plénière du groupe archéologie du handicap (ARC) du 8 mars 2022, au siège de l'Inrap (projet @gir R134932) ;

– réunion plénière des réseaux référents SIG et photogrammétrie le 16 juin sur le site des Mastraits (**Fig. 14**) ;

– à l'occasion d'une visite du chantier par la DRAC Ile-de-France (**Fig. 15**), les travaux sur la base de données ont été présentés à Laurent Roturier (directeur général de la DRAC Ile-de-France), Olivier Peyratout (directeur adjoint délégué, chargé des patrimoines), Nicolas Girault (responsable territorial des dossiers d'archéologie, SRA Ile-de-France) ;

– animation d'un stand sur les outils numériques et l'enregistrement archéologique aux Journées Européennes de l'Archéologie les 17, 18 et 19 juin à Noisy-le-Grand (**Fig. 16**) ;

– présentation des réflexions génératrices de la base de données à la Summer School ARIADNE plus / Cidoc-CRM à Prato, du 20 au 24 juin 2022 (**Fig. 17**) (https://ariadne-infrastructure.eu/mapping-existing-datasets-to-cidoc-crm-summer-school/) ;

– communication au colloque de l'AFAM au Musée de Saint-Germain-en-Laye le 7 octobre 2022 (**Fig. 18**) (https://archeonec.hypotheses.org/afam2022) ;

Fig. 14 Visite des référents SIG lors de la réunion pléniaire du 16 juin 2022

Fig. 15 Visite de la DRAC le 17 juin 2022

Fig. 16 Animation de l'atelier sur les outils numériques et l'enregistrement archéologique

Fig. 17 Présentation de la base de données lors de la Summer School ARIADNE plus/Cidoc-CRM le 21 juin 2022

Fig. 18 Présentation tenue au coloque de l'AFAM le 7 octobre 2022 (St-Germain-en-Laye)

– article dans le hors-série d'Archéopages pour les 20 ans de l'Inrap ;

– dépôts des comptes rendus annuels sur HAL (en cours) (https://hal.archives-ouvertes.fr/hal-03620076) .

5. Perspectives 2023

L'année 2023 doit permettre l'aboutissement du travail entamé ainsi que sa publication.

Dans un premier temps, il conviendra de finaliser le projet Qgis, en affinant et résolvant les dernières difficultés rencontrées en particulier au sein des formulaires.

Les différentes tables de la base de données, que ce soit dans Badass ou dans OTD, ne devraient plus connaître de modification. Il reste cependant des points structurels toujours problématiques dans l'extension funéraire, qui restent à résoudre. Cette nouvelle version sera expérimentée lors de la campagne de fouille programmée de 2023.

L'équipe de recherche sera renforcée par la participation de collègues supplémentaires (Camille Colonna, anthropologue en CIF et Camille Mangier, topographe, référente SIG et photogrammétrie en NAOM) afin de confronter nos choix méthodologiques et de mener à bien nos expérimentations.

Le plugin Qgis devrait également être finalisé. Les options telles que la modification de la base de données existante par l'ajout de nouvelles tables (ex : ajout des tables de l'extension OTD sur une base de données Badass) devraient être intégrées. Ceci devrait également modifier le projet Qgis en permettant d'appeler les tables nouvellement créées et de les afficher dans le panneaux des couches, avec les formulaires adaptés.

Des tests de traitements de modélisation entre différents logiciels de photogrammétrie sont à mener.

Des présentations, dans la continuité de ce qui a été réalisé en 2022, seront également effectuées. Nous pensons notamment présenter la partie Badass, associée au projet Qgis lors des journées archéologiques de la Région Centre dans les années qui viennent,lorsque la version de l'extension QGIS sera stabilisée.

Une publication de synthèse plus globale sur cette expérimentation de dématérialisation d'acquisition des données devra également être effectuée. Un retour d'expérimentation, les méthodologies utilisées (photogrammétrie, SIG) et les protocoles mis en œuvre seront détaillés, ainsi que le processus général de fonctionnement de la base de données, mais également les principes retenus dans son développement.

La documentation générale de la base de données et ses principes de fonctionnement devront être finalisés. Ces fiches techniques ou "manuel d'utilisation" sont destinées à faciliter la prise en main de l'outil dans Qgis. Ce travail de rédaction est en cours mais nécessite l'aboutissement des formulaires afin que cette documentation puisse être accompagnée d'illustrations.

6. Références

6.1 Références bibliographiques en relation avec le projet

Bolo et al. 2014
Bolo A., De Muylder M., Font C., Guillemard T., « De la tablette PC à la cartographie de terrain : exemple de méthodologie sur le chantier d'archéologie préventive de Noyon (Oise) », in *3e Journée d'Informatique et Archéologie de Paris, 1 et 2 juin 2012*, s.l. : s.n., coll. « Archeologia e calcolatori », supplément n°5, pp. 247-256.

Colleter et al. 2020
Colleter R., Romain J.-B., Barreau J.-B., « HumanOS: an open source nomadic software database for physical anthropology and archaeology », *Virtual Archaeol. Rev.*, 11, pp. 94-105.

Delozanne Christel et al. 2021
Delozanne Christel, Seng Christelle, Séguier Jean-Marc, Paccard Nathalie, *Meaux (Seine-et-Marne), 4, rue Camille Guérin : rapport de diagnostic*, Pantin : Inrap CIF.

Delozanne Christel et al. 2022
Delozanne Christel, Pilon Fabien, Séguier Jean-Marc, André Marie-France, Barenghi Frédéric, El Hajaoui Rachid, Quadry Malik, Souffi Bénédicte, *Meaux (Seine-et-Marne), 57 rue Alfred Maury : rapport de diagnostic*, Pantin : Inrap CIF.

Desachy 2008
Desachy B., *De la formalisation du traitement des données stratigraphiques en archéologie de terrain*, phdthesis (Paris, Université Paris 1-Panthéon Sorbonne).

Fournier Laurent et al. 2022
Fournier Laurent, Coussot Céline, Guillemard Thomas, Morisse Valentin, Viale Alexandre, *Eure-et-Loir, Hanches, rue des Prés, rue de la Barre : ZAC Coeur de ville tranche 2 : rapport de diagnostic*, Pantin : Inrap CIF.

Galinié, Randoin 1987
Galinié H., Randoin B., « Enregistrement et exploitation des données de fouilles à Tours », in *Enregistrement des données de fouilles urbaines*, Centre National d'Archéologie Urbaine, vol. 1, Tours : s.n., pp. 61-73.

Kasser 2001
Kasser Y. E. Michel, *Digital Photogrammetry*, London : CRC Press.

Le Forestier 2019
Le Forestier C., « Rapport de fouilles Les Mastraits, Noisy-le-Grand (93) », *Archéologie des nécropoles* [en ligne], URL : https://archeonec.hypotheses.org/959 [lien valide au 23 juin 2023].

Le Forestier Cyrille et al. 2012
Le Forestier Cyrille, Lafarge Ivan, Parot Sabrina, Perrier Daniel, *Noisy-le-Grand (Seine-Saint-Denis), 4 rue des Mastraits : [rapport de fouille]*, Epinay-sur-Seine : Service départemental d'archéologie de Seine-Saint-Denis.

Lozano, Georges 2019
Lozano V., Georges É., « Not Only SQL », *Framabook* [en ligne], URL : https://archives.framabook.org/not-only-sql/ [lien valide au 14 avril 2023].

Mercey Florent et al. 2022
Mercey Florent, Millet Sébastien, Guibert Patrick, *Sully-sur-Loire, Loiret, la Brosse 2 : rapport de diagnostic*, Pantin : Inrap CIF.

Moreau 2016
Moreau A., « GIS, An Answer to the Challenge of Preventive Archaeology? The Attempts of the French National Institute for Preventive Archaeology (Inrap) », *CAA2015 Keep Revolut. Going Vol. 1 Proc. 43rd Annu. Conf. Comput. Appl. Quant. Methods Archaeol. Siena Italy Mar 31 St - Apr 2 Nd 2015*, 1, pp. 303-308.

Pierrot-Deseilligny, Clery 2011
Pierrot-Deseilligny M., Clery I., « Evolutions récentes en photogrammétrie et modélisation 3D par photo des milieux naturels », *Collect. EDYTEM Cah. Géographie*, 12, 1, pp. 51-66.

Rose 2012
Rose H., « Processus de création de nuages de points par corrélation d'images ».

Seng et al. 2020
Seng C., El-Hajaoui R., Font C., Guillemard T., Le Forestier C., Florent M., *La Nécropole Numérique - Compte-rendu d'activité - 2020* [en ligne], Intern report, s.l. : Inrap, URL : https://hal.science/hal-03620076.

Seng et al. 2022
Seng C., Le Forestier C., El-Hajaoui R., « Acquisition numérique des données archéo-anthropologiques et spatiales - Le projet « Nécropole Numérique » », *Archéopages*, 2, 6, pp. 328-333.

Seng Christelle et al. 2023
Seng Christelle, Bauchet Olivier, Bustos Caroline, Chaussé Christine, Le Goff Céline, Lepareux-Couturier Stéphanie, Paccard Nathalie, Delozanne Christel, *Meaux (Seine-et-Marne), Cité de la Musique Simone Veil : rapport de diagnostic*, Pantin : La Courneuve.

6.2 Sitographie

Baiet [sans date]
Baiet, « Plugin Badass QGIS » [en ligne], URL : https://plugins.qgis.org/plugins/badass/ [lien valide au 23 juin 2023].

El-Hajaoui [sans date]
El-Hajaoui R., « Archéologie des nécropoles » [en ligne], URL : https://sketchfab.com/ArcheoNec [lien valide au 23 juin 2023].

Guillemard [sans date]
Guillemard T., « La base de données CADoc » [en ligne], URL : https://sites.google.com/site/bdcadoc/ [lien valide au 23 juin 2023].

[Sans date]
« Bibliothèque de ressources pédagogiques de l'ENSG École Nationale des Sciences Géographiques » [en ligne], URL : http://cours-fad-public.ensg.eu/course/index.php?categoryid=43 [lien valide au 23 juin 2023].

6.3 Logiciels et softs utilisés

- Agisoft Metashape Pro
- AutoCAD Map
- QGIS (https://www.qgis.org/fr/site/)
- spatialite.gui (https://www.gaia-gis.it/fossil/spatialite_gui/index)
- draw.io
- dbdiagram.io
- notepad++ (https://notepad-plus-plus.org/)
- QT Creator (https://www.qt.io/)
- SQLite for dbBrowser (https://sqlitebrowser.org/)
- Cloud Compare (Daniel Girardeau-Montaut et EDF R&D, cloudcompare.org)
- image app (Panasonic)7
- advanced renamer

7. Annexes

7.1 Rapport de stage d'Emma Chapuis

DÉVELOPPEMENT DU PROJET BADASS

Rapport de stage de L3 Informatique - ISTIC - Université Rennes1

Emma CHAPUIS
Année 2020 – 2021
Du 26 avril 2021 au 18 juin 2021
Université de Rennes1 – UFR SPM au CReAAH
Tuteur de stage : M. Jean-Baptiste Barreau
Enseignante Référente : Mme. Véronique MASSON

Remerciements

Je suis très reconnaissante envers Mme. FONT, Mme. SENG, M. GUILLEMARD et M. MERCEY, archéologues à l'origine du projet, pour leur réactivité et leurs réponses à mes questions.

Je remercie M. BARREAU, ingénieur informatique au CReAAH et tuteur dans le cadre de ce stage, pour m'avoir permis de réaliser ce stage et pour m'avoir encadré pendant le processus de développement.

Je tiens également à remercier M. HUMEAU, le stagiaire qui a commencé le développement du projet, pour son aide et ses conseils.

Enfin merci à Mme. MASSON, la responsable de la L3 Informatique, qui en ces temps troublés a porté une grande assistance pour la gestion des conventions de stage.

Table des matières

1. Le cadre du stage

Ce stage s'est déroulé au sein du CReAAH, un laboratoire de recherche situé à Rennes. Le CReAAH regroupe l'Université de Rennes 1, l'Université de Rennes 2 ainsi que l'Université de Nantes et l'Université du Mans, le tout sous la tutelle du CNRS et en partenariat avec l'INRAP. Il réunit donc les archéologues, historiens, archéomètres et paléoenvironnementalistes du grand quart nord-ouest de la France. Le centre est actuellement sous la direction de Marie-Yvane Daire, secondé de Rita Soussignan, son adjointe. Chaque université est sous la direction d'un représentant, mais la majorité des services sont communs aux différents sites, c'est notamment le cas pour les services de l'administration, de la finance, de la documentation et valorisation, mais aussi de l'hygiène et de la sécurité, ainsi que du service informatique, modélisation 3D, BDD et SIG, service dans lequel se déroule ce stage. Ce service est composé de deux personnes, et leurs principales activités sont, le développement et maintenance d'applications, de bases de données pour l'exploitation, la gestion et le stockage des données scientifiques. Ainsi que la modélisation et numérisation 3D de sites et objets archéologiques.

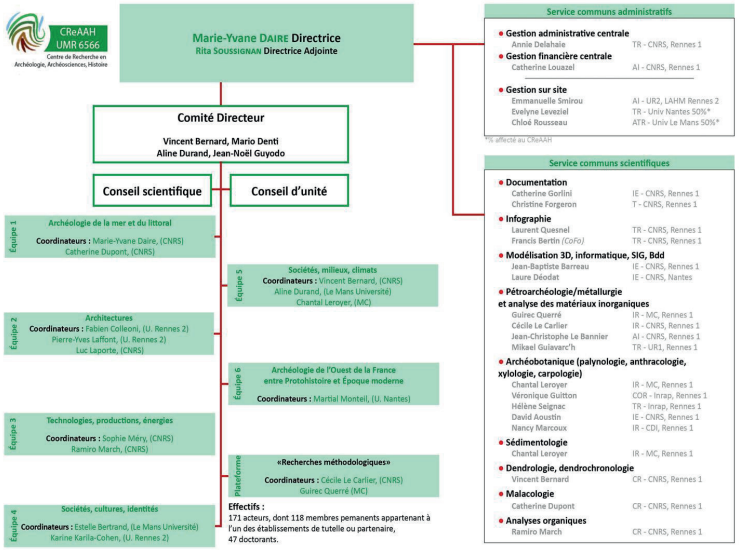


Figure 1 - Organigramme général du CReAAH Sources : Le site du CReAAH

2. La problématique

2.1. L'outil QGIS

QGIS, anciennement appelé Quantum GIS, est un logiciel SIG (Système d'Information Géographique) libre, multi-plateforme. Il est écrit en C++ et en python, et utilise Qt comme support graphique. Le logiciel est grandement répandu de par sa gratuité, les nombreuses langues disponibles, mais également de par les nombreux aspects pris en charge, que ce soit les couches matricielles ou les couches vectorielles par exemple. Un autre de ses points forts est la possibilité de stocker les données utilisées dans une base de données, avec un large choix de systèmes de gestion de bases de données tels que PostgreSQL, Oracle, MySQL, DB2 ou SQLite (c'est ce dernier qui a été retenu ici).

QGIS offre beaucoup de fonctionnalités qui font de lui un logiciel très utilisé par les archéologues. Il est d'ailleurs soutenu par la fondation OSGeo (Open Source Geospatial), et est recommandé par l'État français. La principale fonctionnalité qui nous intéresse dans le cadre de ce projet est celle de pouvoir réaliser des plugins pour QGIS. Ses extensions sont réalisées en python ou en C++. Dans le cadre de ce projet, c'est le python qui a été choisi.

2.2. Le domaine d'application et les tâches demandées

Ainsi, le stage a entièrement consisté au développement d'une partie d'un plugin en langage Python. Ce plugin, nommé BADASS (Base Archéologique de Données Attributaires et Spatiales), vient d'une volonté de dématérialiser l'enregistrement des données de terrain et s'inscrit dans le cadre du développement d'une série d'outils et de logiciels au sein de l'Inrap, en concordance avec la mutation numérique actuellement en cours. Son but est l'enregistrement des unités stratigraphiques composant chaque sépulture, ainsi que les données propres à l'archéo-anthropologie au sein d'une même base de données.

Pour préparer mes débuts sur le projet, je me suis familiarisée avec le logiciel QGIS en effectuant des tutoriels touchant à tous les aspects de l'outil. Ensuite mon rôle principal a été de commencer à développer l'extension OTD (Of The Dead) de BADASS servant à rentrer les données relatives aux squelettes trouvés sur un site archéologique. Enfin, une fois M. HUMEAU, le premier stagiaire, partit, j'ai poursuivi le code principal du plugin là où il l'avait laissé (le code principal concernant l'enregistrement des données générales d'un site archéologique).

3. Le contenu

3.1. Partie I : les tutos

Comme mentionné plus haut, pour me préparer à travailler sur le projet et me familiariser avec l'outil QGIS on m'a demandé de faire tous les tutoriels présents sur http://www.qgistutorials.com/fr/ .

Ces tutoriels vont de l'utilisation de l'outil au développement de plugins qui lui sont destinés.

3.2. Partie II : début de l'extension OTD

Après m'être familiarisée avec l'outil, on me demanda de commencer à travailler sur l'extension OTD (Of The Dead) du plugin BADASS. J'ai commencé par regarder ce que M. HUMEAU, l'autre stagiaire sur le projet, avait commencé à faire pour le code principal du plugin. M. HUMEAU avait à sa disposition les fenêtres pour la partie principale du plugin sous forme de fichier .ui ainsi que le fichier SQL servant à créer les tables voulues dans la base de données créée en même temps qu'un projet via ce plugin. Il a développé la base du plugin permettant d'ajouter le bouton avec son icône dans la barre d'outils et d'ouvrir la fenêtre d'accueil (cf. figure 2) à l'appui sur ce bouton. Il a aussi ajouté les liens entre les diverses fenêtres pour provoquer leur ouverture lorsque le bouton correspondant était enfoncé. En plus de ça, il a créé toutes les fonctionnalités nécessaires pour la fenêtre principale du projet (cf. figure 3), pour la fenêtre des ouvertures (cf. figure 4) et pour les autres fenêtres liées à la fenêtre des ouvertures. Pour la fenêtre principale du projet, les fonctionnalités incluent la création d'une base de données, et donc d'un projet, et la création de toutes les couches, vectorielles ou non, liées à cette base de données (car il ne faut pas oublier l'aspect graphique de l'outil qui permet d'enregistrer des données géométriques).

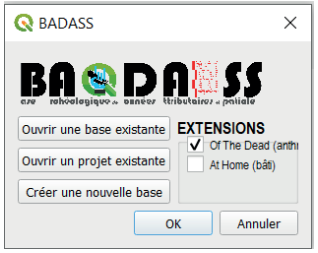


Figure 2 - fenêtre d'accueil de BADASS

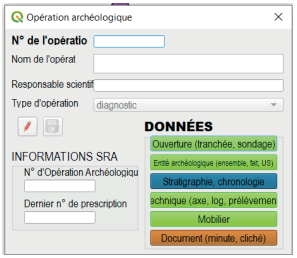


Figure 3 - fenêtre principale du projet

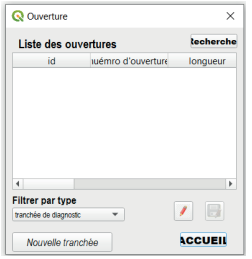


Figure 4 - fenêtre des ouvertures

Comme les besoins n'étaient pas encore clairement définis pour l'extension OTD, j'ai demandé aux archéologues travaillants sur le projet ce qu'ils voulaient pour cette extension et on m'a fourni un fichier SQL propre à cette extension car l'extension pouvant être activée ou non, les tables y correspondant

étaient définie à part. Dans un premier temps j'ai donc développé de quoi ajouter les tables du fichier SQL fournit lors de la création d'une nouvelle base de données (et donc d'un nouveau projet) si la case de l'extension était cochée. Cela rajoutait donc dans le projet les couches correspondant aux données liées aux squelettes comme celle que l'on peut voir dans la capture à la figure 5. Après quelques essais et des réflexions sur comment enregistrer les données du squelette, il a été défini que je devais utiliser un outil de l'application QGIS permettant d'ajouter une action à une couche vectorielle qui se déclenchait lors de la sélection d'un élément de cette couche (cette action se définissait normalement dans les paramètres via l'application et il fallait que j'automatise la création de l'action dans la couche voulue en code python). Je devais aussi créer moi-même des formulaires au format .ui pour l'enregistrement des données du squelette et faire en sorte que ces formulaires s'ouvrent en cliquant sur un élément d'une couche correspondant au squelette. Il y a trois couches : une avec le squelette entier (cf. figure 5), une avec les différentes parties anatomiques (cf. figure 7) et une avec tous les os (cf. figure 8). Comme demandé, j'ai créé les formulaires (cf. figures 6, 7 et 8), les actions permettant de les lancer, ai permis l'enregistrement des données rentrées dans le formulaire dans la table correspondante et, pour faciliter l'entrée des données, ai automatisé le remplissage des champs pour lesquels les données pouvaient être récupérées lors de la sélection de l'élément sur la couche vectorielle (pour ce dernier point, la couche vectorielle étant liée à la base de données, il est possible de récupérer avec certaines fonctions les données de l'élément sélectionné). Aussi, j'ai mis en place l'auto incrémentation de l'ID destiné à la base de données dans le formulaire. Après ça, après concertation avec l'équipe d'archéologues, j'ai rajouté pour chacune des 3 couches une action et un formulaire permettant de modifier l'élément sélectionné en choisissant dans le formulaire le squelette concerné. Lors de la sélection du squelette, l'entrée de la table concernée est retrouvée automatiquement (grâce au numéro de squelette et aux numéros de parties anatomiques qui permettent de faire le lien entre les tables) et les données déjà enregistrées sont affichées dans le formulaire pour le confort de l'utilisateur (cf. figures 9, 10 et 11).

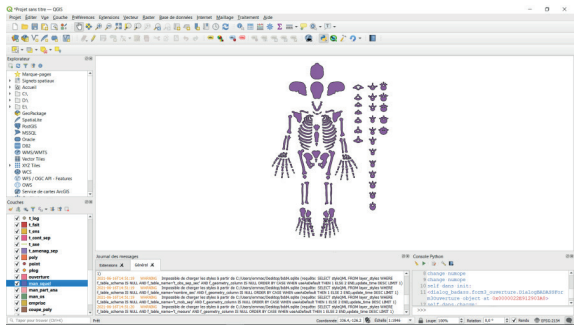


Figure 5 – la couche correspondant au squelette entier de l'extension OTD

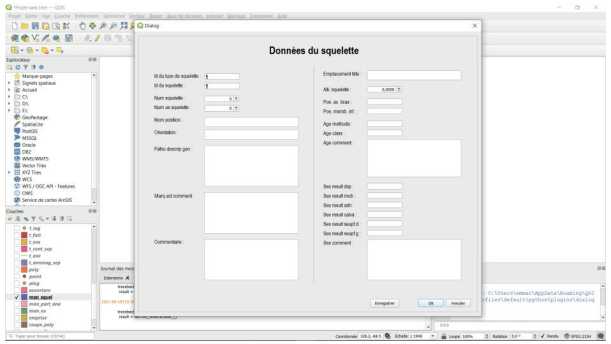


Figure 6 - la fenêtre affichée en cliquant sur le squelette dans la couche de la figure 5

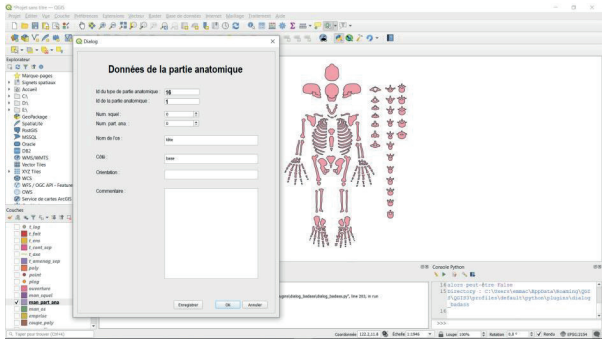


Figure 7 - la couche des parties anatomique et la fenêtre ouverte en cliquant sur le crâne

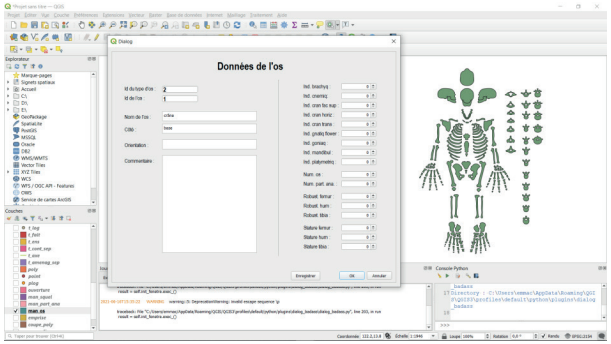


Figure 8 - la couche des os et la fenêtre ouverte en cliquant sur le crâne

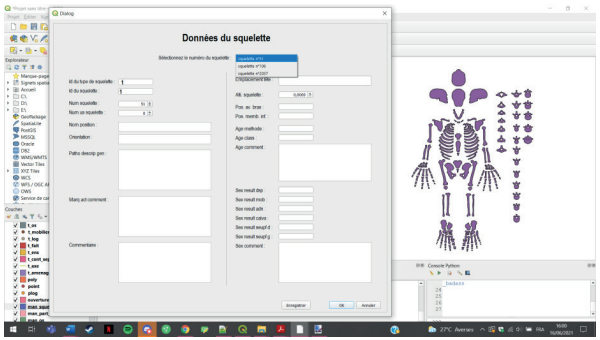


Figure 9 - la couche correspondant au squelette et le formulaire de modifications

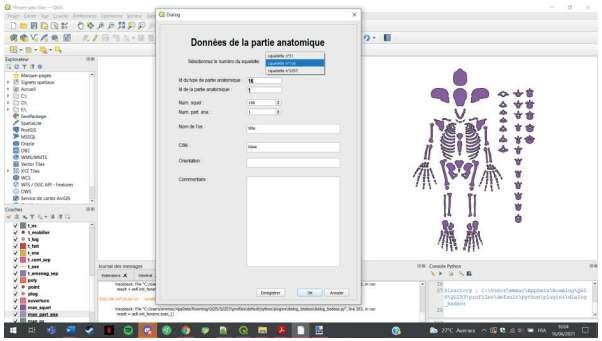


Figure 10 - la couche des parties anatomique et le formulaire de modifications

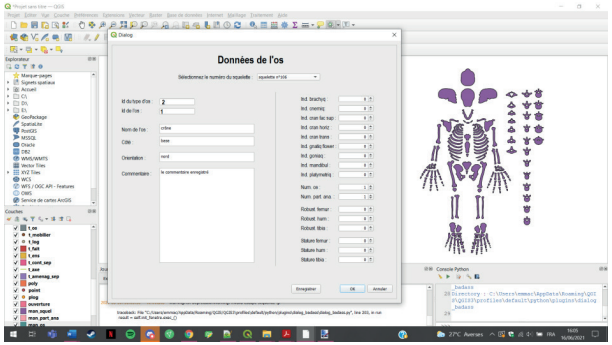


Figure 11 - la couche des os et le formulaire de modifications

Note 1 :

Toute cette partie a été testée sur le terrain d'après les archéologues et il reste des ajustements à faire mais je n'ai pas eu d'échos de dysfonctionnements.

Note 2 :

Voici où se fait la sélection de l'action que l'on veut effectuer en cliquant sur un élément d'une couche :

3.3. Partie III : continuer le code principal du plugin

Après avoir effectué cette partie, j'ai été en charge de continuer le code de la partie principale du plugin là où l'autre stagiaire l'avait laissé. C'est-à-dire que j'ai dû implémenter les fonctionnalités de la fenêtre des entités (cf. figure 12, déjà commencée par M. HUMEAU) et des formulaires en dépendant (cf. figure 13 pour exemple). Il était principalement question de permettre l'enregistrement des données des différentes entités dans les formulaires dépendants de la fenêtre des entités.

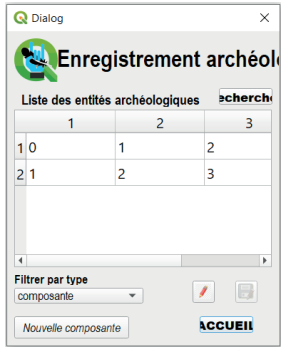


Figure 12 - la fenêtre des entités

Figure 13 - le formulaire des ensembles qui dépend des entités

4. Bilan

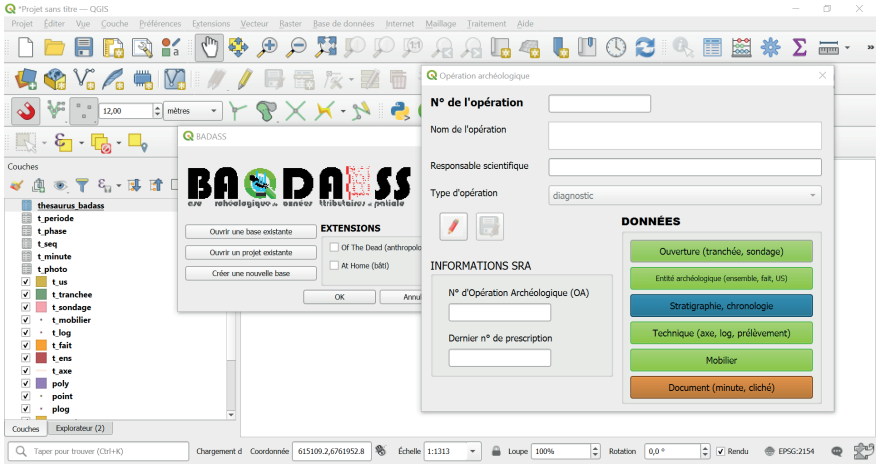
Le plugin BADASS a été bien entamé par M. HUMEAU et moi, mais vu l'envergure du projet il nous a été impossible de le mener jusqu'au bout. Il restera à compléter le code principal du plugin, développer sa deuxième extension, corriger les bugs qui pourraient être trouvés après plus de tests et mettre en place les ajustements demandés par les archéologues sur le projet.
Ce projet a nécessité beaucoup d'autonomie pour 2 causes principales :

- Le distanciel, qui compliquait la communication
- La vision du projet qui, à l'échelle globale, est bien établie mais qui, au niveau des spécifications, manque de détails du fait de la récence du projet et de la nécessité de retours de professionnels travaillant sur le terrain

Dans cette même veine, une des difficultés du projet a été de tester l'application car les tests ne pouvaient être complets que dans le cadre dans lequel allait être utilisée l'application.
Aussi, la résolution des erreurs que j'ai pu rencontrer en développant a été assez problématique du fait de la spécificité des fonctions à utiliser pour un plugin destiné à QGIS. Bien qu'il y ait une documentation, j'ai trouvé la plupart des informations dont j'avais besoin non pas dans celle-ci mais sur les forums.
En bref, j'ai eu beaucoup de difficultés sur ce projet mais cela a été très formateur comme expérience de développement et dans l'apprentissage du Python dont je ne connaissais que les bases.

7.2 Rapport de stage d'Alexandre Humeau

RAPPORT DE STAGE



Développement en Python d'une interface de saisie et gestion de données

Du 06 Avril 2021 au 28 Mai 2021

Alexandre HUMEAU
Étudiant 2ème année DUT Réseaux & Télécoms

Jean-Baptiste BARREAU - Tuteur entreprise
Jean-Claude NUNES - Tuteur IUT

Établissement de formation :
IUT de Saint-Malo
Rue de la Croix Désilles
CS 51713
35417 Saint-Malo Cedex

Structure d'accueil :
UMR 6566 CReAAH
Campus de Beaulieu, Bâtiment 25
Labo Archéosciences
Avenue du Général Leclerc – CS 74205
5042 Rennes Cedex – France

RAPPORT DE STAGE

Développement en Python d'une interface de saisie et gestion de données

Du 06 Avril 2021 au 28 Mai 2021

Alexandre HUMEAU
2ème année DUT Réseaux & Télécoms

Jean-Baptiste BARREAU - Tuteur entreprise

Jean-Claude NUNES - Tuteur IUT

Établissement de formation :
IUT de Saint-Malo
Rue de la Croix Désilles
CS 51713
35417 Saint-Malo Cedex

Structure d'accueil :
UMR 6566 CReAAH
Campus de Beaulieu, Bâtiment 25
Labo Archéosciences
Avenue du Général Leclerc – CS 74205
5042 Rennes Cedex – France

REMERCIEMENTS

Je tiens à remercier Monsieur Jean-Baptiste Barreau, ingénieur informatique au CReAAH* et tuteur dans le cadre de ce stage, pour m'avoir permis de réaliser ce stage et pour m'avoir encadré pendant le processus de développement. Ainsi que pour avoir répondu à mes interrogations afin de mener à bien la réalisation de ma partie du projet.

Je remercie également Monsieur Jean-Claude Nunes, professeur à l'IUT de Saint-Malo et tuteur IUT dans le cadre de ce stage, pour son encadrement et son investissement. Ainsi que toute l'équipe pédagogique et plus généralement tous le personnel de l'IUT pour les deux années.

J'exprime toute ma reconnaissance à Monsieur Florent Mercey, Madame Caroline Font, Monsieur Thomas Guillemard et Madame Christelle Seng, archéologues à l'origine du projet, pour m'avoir éclairé quant aux attendus du projet. Ainsi que pour leur disponibilité et réponses quant à mes questionnements. Mais également pour m'avoir fait entrevoir le milieu de l'archéologie.

Et pour finir, merci à Madame Emma Chapuis, autre stagiaire présente dans le développement du projet, pour son aide et ses conseils.

Sommaire

1 Introduction

Les données sont l'information. Elles sont partout et pourtant nous n'y fessons pas attention, car trop souvent brutes et vide de sens. Elles murmurent aux oreilles des scientifiques, et ils les chérissent pour qu'elles racontent leurs secrets. Elles sont l'un des pilier de la recherche, mais leur quantité peut aussi bien la faire s'effondrer. Pour éviter cela, elles sont minutieusement choisies, triées et stockées, de manière physique, puis numérique depuis la révolution numérique. La numérisation des données permet un plus simple accès, mais également un plus simple partage. Les données sont plus simplement consultables et modifiables.

C'est dans le cadre de cette mutation numérique que l'Inrap* à lancé le développement d'une série d'outils, afin de dématérialiser l'enregistrement des données de terrain. Un de ces outils est réalisé par un groupe d'archéologues, avec le soutien de Jean-Baptiste Barreau, ingénieur informatique. Cet outil doit permettre d'enregistrer les unités stratigraphiques* composant chaque sépulture, ainsi que les données propres à l'archéo-anthropologie au sein de la même base de données. La saisie et l'interrogation de ces données se feront par l'intermédiaire d'une extension au logiciel QGIS*. Pour davantage d'ergonomie et d'accessibilité, les données seront affichées sous forme de formulaires.

C'est dans ce contexte que se déroule ce stage. À partir du travail déjà réalisé, commence le développement de l'extension, en suivant les directives afin de répondre au mieux aux attentes. Ce stage c'est déroulé au sein du CReAAH, un laboratoire de recherche situé à Rennes. Ce laboratoire est sous la tutelle de l'université de Rennes 1 et Rennes 2, ainsi que de l'université de Nantes et du Mans. Mais également du ministère de la culture et du CNRS*, organisme auquel appartient Jean-Baptiste Barreau, maître de stage. L'Inrap est également un établissement partenaire. Le centre est sous ce forma depuis 1991, c'est est une unité largement interdisciplinaire, à la croisée des sciences humaines et sociales, des sciences de l'environnement et des sciences physiques et chimiques.

Durant mon stage, j'ai donc participé au développement en Python* d'une interface de saisie et de gestion de données, à partir des outils déjà établis. L'enjeu était de réussir à mettre en place une extension qui soit simple d'utilisation, mais complète, afin d'offrir une solution qui répond aux attentes.

Dans un premier temps nous palerons de l'organisme d'accueil. Avec une présentation de son histoire, de son organisation et de ses activités, ainsi qu'une rapide présentation du CNRS et de l'Inrap. Puis dans un second temps, nous introduirons le projet, avec ces origines, les outils déjà présent au début du stage et les concepts importants pour sa réalisation. Et pour finir, nous parlerons du travail réalisé, au travers de la construction des différents formulaires. Ainsi que du bilan et des perspectives du projet.

2 Introduction in english

The data are the information. They are everywhere and however, we don't look at her, because they are often crude and meaningless. They whisper in the ears of the scientists, and they cherish them so that they tell their secrets. They are one of the pillar of the recherche, but their quantity can made it fall. To avoid it, they are choose with attention, select and store, physically, then digitally since the digital revolution. The digitization of the data offered a simple access, and a simple share. The data are more simply available for consultation and modification.

It is in this digital transformation, that the Inrap start the development of multiple tools, to dematerialize this field data saved. One of them is made by a group of archaeologists, with the support of Jean-Baptiste Barreau, computer engineer. This tool will be able to saved the archaeological data in database. The addition, consultation and modification of the data will be made by a plugin of QGIS software. For greater ergonomics and accessibility, the data will be displayed in the form of forms.

It is in this context that this internship takes place. From the work done, start the plugin development, by following the recommendation, to respond at the expectations. This internship happened in the CReAAH, a research laboratory located in Rennes. This laboratory is under the supervision of the university of Rennes 1 and Rennes 2, as well as the university of Nantes, and Le Mans. But also under the minister of the culture and the CNRS, establishment which Jean-Baptiste Barreau, internship master, come from. The Inrap is also a partner establishment. The CReAAH is under this form since 1991, it's a laboratory multi-discipline, at the crossroads of the human and social sciences, environment sciences and physical and chemical sciences.

During my internship, I work on python development of a data management interface, from the given tools. The aim was to make a plugin which is simple to use, but with full option, to response at the expectation.

Firstly, we are going to talk about the CReAAH, the structure who welcomed me. With a presentation of her history, her organization, and her activity. Also a little presentation of the CNRS and the Inrap. Secondly, we are going talk about the project, where he come from, what he is for, the tools already made and the important concepts. Finally, we are going to present the work done, with the structure of the different forms. We are also going to talk about the actual result of this project, and its future.

3 Le CReAAH, un laboratoire à la croiser des domaines

3.1 Le CReAAH d'hier à aujourd'hui

Pour revenir aux origines de l'UMR* 6566 CReAAH ou l'Unité Mixte de Recherche 6566, Centre de Recherche en Archéologie, Archéosciences*, Histoire, il faut remonter l'immédiat après-guerre. C'est à cette époque qu'intervient la création du Laboratoire d'Anthropologie de l'Université de Rennes. Par la suite celle de l'équipe de Recherche n°27 du CNRS à la fin des années 60, qui se transforme en 1988, en Unité propre de Recherche, l'UPR* 403 du CNRS.

Puis, c'est à la suite l'association des laboratoires d'Anthropologie et d'Archéométrie (physique) de l'Université de Rennes 1 et du laboratoire d'Archéologie-Histoire de l'Université de Rennes 2, que l'UPR se transforme pour devenir l' UMR 153, en 1991. Et devient par la suite l' UMR 6566 C2A, pour Civilisations Atlantiques et Archéosciences. Avec l'intégration des personnels du CNRS, mais également des Universités de Rennes 1 et Rennes 2, ainsi que du Ministère de la Culture, par la suite de l'Université de Nantes, et plus récemment de l'INRAP* et de l'Université du Maine.

Et c'est en 2008 que le laboratoire acquière la dénomination sous laquelle il est aujourd'hui connu. L'URM 6566 C21 devient alors l'URM 6566 CReAAH, ce changement de nom intervient pour marquer l'entrée de nombreux collègues historiens, des universités de Rennes 2 en 2008, et du Mans en 2010. La figure 1 reprend tous ces éléments sous la forme d'un diagramme.

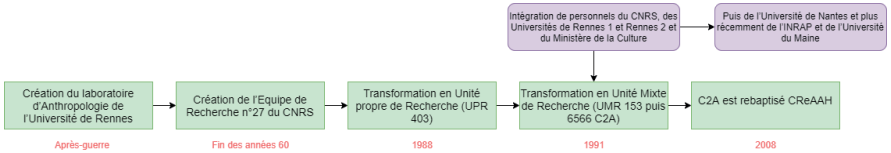


Figure 1: Diagramme représentant l'histoire du CReAAH

3.2 Plusieurs sites, mais une organisation commune

Comme énoncé précédemment, le CReAAh regroupe l'Université de Rennes 1, l'Université de Rennes 2 ainsi que l'Université de Nantes et l'Université du Mans, le tout sous la tutelle du CNRS et en partenariat avec l'INRAP. Il réunit donc les archéologues, historiens, archéomètres et paléoenvironnementalistes* du grand quart nord-ouest de la France. Le centre est actuellement sous la direction de Marie-Yvane Daire, secondé de Rita Soussignan, son adjointe. Chaque université est sous la direction d'un représentant, mais la majorité des services sont communs aux différents sites, c'est notamment le cas pour les services de l'administration, de la finance, de la documentation et valorisation, mais aussi de l'hygiène et de la sécurité, ainsi que du service informatique, modélisation 3D, BDD* et SIG*, service dans lequel se déroule ce stage. Ce service est composé de deux personnes, et leurs principales activités sont, le développement et maintenance d'applications,

de bases de données pour l'exploitation, la gestion et le stockage des données scientifiques. Ainsi que la modélisation et numérisation 3D de sites et objets archéologiques.

Outre ces différents services, le centre est également composé de différents conseils, comme le conseil de direction qui regroupe les différents représentants de site ainsi que la directrice et son adjointe, le conseil de scientifique qui regroupe les différents responsables d'équipe et de plateforme, ou encore le conseil d'unité. Avec pour chacun des conseils des rôles décisionnels, afin de garantir la cohérence des projets et actions menées par le centre.

Le centre est avant tout un centre de recherche, nous y retrouvons actuellement 6 équipes travaillant chacun, sur des projets dans des spécialités différentes, allant de l'archéologie à l'environnement, en passant par les sciences humaines et sociales. Et une septième portée sur la méthodologie.

La Figure 2 regroupe toutes les composantes du centre sous la forme d'un organigramme. À noter que l'effectif actuel du centre est de 171 acteurs, donc 118 membres permanents et 47 doctorants.

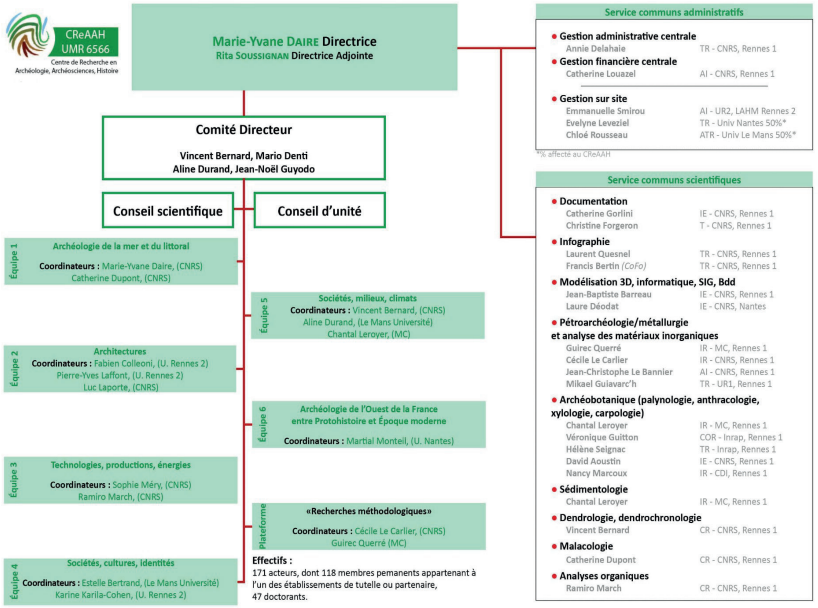


Figure 2: Organigramme général du CReAAH Sources : Le site du CReAAH

Les 6 équipes de l'URM 6566, travailles toutes sur des sujets dans des domaines différents, allant des sciences humaines et sociales, jusqu'aux sciences physiques et chimiques, en passant par les

sciences de l'environnement. Mais tous ont une discipline en commun, l'archéologie. Les projets sur lesquels travail les équipes, sont réalisés à l'initiative du centre, mais aussi en partenariat, avec d'autres centres de recherche, en France ou à l'international. Chaque équipe travail sur plusieurs projets, qui donne lieu à plusieurs thèses. Ces thèses sont les principaux résultats des actions menées, et obtiennent parfois des parutions dans des revues scientifiques, voire des récompenses.

3.3 Des partenaires de confiance

Comme énoncé précédemment, le CReAAh est sous la tutelle du CNRS, et en partenariat avec l'Inrap. C'est d'ailleurs cet institut, dont sont originaires les archéologues qui travaillent sur ce projet.

Le CNRS, est le plus grand organisme public français de recherche scientifique. Il a été fondu le décret-loi du 19 octobre 1939. En vue de coordonner les activités des laboratoires, pour obtenir un meilleur rendement de la recherche scientifique. Il est aujourd'hui placé sous la tutelle administrative du ministère de l'Enseignement supérieur, de la Recherche et de l'Innovation. Avec un budget de 3,4 milliards d'euros, il rassemble plus de 32 000 personnes dans plus de 1100 laboratoires en France et à l'étranger.

Il exerce son activité dans tous les domaines de la connaissance, dix instituts nationaux spécialisés dans un domaine de la connaissance (sciences humaines et sociales, biologie, chimie, écologie et environnement, sciences de l'information, sciences de l'ingénierie et des systèmes, mathématiques, physiques, physique nucléaire et des particules, sciences de l'univers). Il est considéré comme le second plus grand centre de recherche au niveau mondial, et le premier au niveau européen.

La mission principale du CNRS est de "Identifier, effectuer ou faire effectuer, seul ou avec ses partenaires, toutes les recherches présentant un intérêt pour la science ainsi que pour le progrès technologique, social et culturel du pays.". Elle a été confiée par l'État, au travers du décret du 24 novembre 1982. Pour mener à bien cette mission, elle a été divisée en cinq axes. Le premier est bien sur celui de la recherche. Le second est celui de valoriser les résultats, afin de les faire profiter à la société. La société profite également des fruits de la recherche au travers de connaissances qui lui sont partagées. Pour tenter d'accomplir le mieux possible, et le plus longtemps possible, former les chercheurs demain est essentiel. Mais il faut également que des moyens importants soient mis en œuvre, le CNRS contribue à la politique scientifique.

L'Inrap est un établissement public, à caractère administratif de recherche français, créé par la loi du 17 janvier 2001 relative à l'archéologie préventive. Il est placé sous la tutelle des ministères chargés de la Culture et de la Recherche.

La principale mission de l'institut est l'archéologie préventive. L'archéologie préventive, étudie et préserve les éléments significatifs du patrimoine archéologique menacés par les travaux d'aménagement. L'institut assure la détection et l'étude du patrimoine archéologique touché par les

travaux d'aménagement du territoire. Les autres missions de l'institut, qui découle de l'archéologie préventive sont, l'exploitation des données récoltée et des résultats obtenus, dans le cadre de recherche, qui peut être menée en partenariat avec des universités, et des centres de recherche national, européen et international. Ainsi, que l'exposition des résultats obtenue, auprès de la population, pour participer à la conservation du patrimoine culturel, et valorisé l'archéologie. Qui s'organise autour de différentes actions, des portes ouvertes sur les sites archéologiques, des ouvrages destinés à tout public, adulte et enfant. Mais aussi l'organisation d'événement lors des journées consacrées à la science en générale, et depuis 2010, l'organisation de journées nationales de l'archéologie.

4 Un projet pour et par les archéologues

4.1 Les origines du projet : la gestion des données numériques

Depuis plus de 10 ans, la nécropole de Noisy-le-Grand(93) a bénéficié de plusieurs campagnes de fouilles préventives. De 2019 à 2021, une fouille* programmée permet la mise au jour de nouvelle découverte, afin d'enrichir les données déjà recueillies. C'est dans ce cadre et avec une volonté de dématérialiser l'enregistrement des données de terrain, que commence le développement d'une série d'outils et de logiciels au sein de l'Inrap, en concordance avec la mutation numérique actuellement en cours.

L'un de ces outils a pour but, l'enregistrement des unités stratigraphiques composant chaque sépulture, ainsi que les données propres à l'archéo-anthropologie au sein de la même base de données. Cet outil, BADASS*, est un plug-in, une extension à QGIS, un logiciel libre très utilisé par les archéologues, afin d'entre autres visualiser, consulter, modifier, ajouter des données géométriques et attributaires. L'outil BADASS répond à un besoin d'ergonomie, de fluidité et de simplicité dans ces actions. En effet, avec QGIS, il est tout à fait possible de faire tout cela, mais ces tâches s'effectuent pour la plu-part au travers d'un tableau. Là où BADASS propose des formulaires, afin d'aider l'utilisateur dans la manipulation de ces données.

Outre les formulaires, l'outil BADASS, va également intégrer un modèle 3D. Ce modèle va permettre d'éditer une orthophotographie, qui remplacera les prises de données manuelles sur le terrain. Ainsi que la possibilité d'archiver les volumes du vestige archéologique, et sa position dans un espace géo-référencé.

4.2 L'existant du projet : de la base de données au visuel

En raison de l'état d'urgence sanitaire lié à la pandémie de Covid-19, les travaux prévus sur le terrain n'ont pas pu avoir lieu. Cette situation a favorisé l'accélération du projet, en augmentant le temps lui étant consacré. C'est dans ce contexte qu'a été organisée la réunion du 18 juin 2020, réunion qui a permis de mettre les choses au claire entre les différents membres du projet. Cette réunion a donné lieu à de nombreux échanges et questionnements, démontrant le travail faramineux

que demande ce projet. Par la suite, d'autres réunions et échanges ont eu lieu dans le but de mettre en place le modèle de la base de données.

4.2.1 La base de données

Les archéologues présents sur le projet ont dans leur rang des personnes expérimentées, avec la création de bases de données destinées à la gestion de données géométrique et attributaire, et plus généralement avec le langage SQL*. La base de données destinée au projet BADASS a donc été conceptualisée par les archéologues, pour répondre au mieux à leurs besoins. Étant donné la grandeur et la complexité de la base de données résultante, il ne sera traité ici que des éléments utilisés dans la partie du projet traité lors de ce stage. À savoir les tables emprise, ouverture, t_sondage et t_tranchee, et une partie des triggers leur étant associer.

Pour ce qui est du choix de la base de données, la technologie choisie doit respecter plusieurs conditions. Elle doit permet la gestion des données spatiales et attributaires, mais également leur interopérabilité et leur pérennité. Deux solutions étaient alors envisagées, PostGis l'interface spatiale de PostgreSql, et Spatialite, l'interface spatiale de SQLite. Le choix final a été celui de Spatialite, en raison de sa simplicité d'utilisation pour l'utilisateur. En effet Spatialite ce présent sous la forme d'un unique fichier, qui peut être facilement déplacé, sauvegardé et conserver par l'utilisateur. En opposition à PostGis, qui demande l'utilisation d'un serveur de base de données avec un hébergement dédié. La contre-partie à la simplicité apportée par Spatialite, est le nombre de fonctionnalités limité, ainsi qu'un stockage inférieur à celui de PostGis. Cela n'est cependant pas un problème étant donné que les volumes de données habituellement manipulés sont inférieurs aux maxima fournis par Spatialite. À savoir 281 tera-octets de données, pour une base de données et 2^{64} enregistrements pour une table.

4.2.2 Les tables et triggers associés au projet

Comme énoncé précédemment, seule la partie de la base de données concernant la partie du projet réaliser dans le cadre de ce stage va être abordée. Cela concerne donc les tables emprise, ouverture, t_sondage et t_tranchee, et une partie des triggers* leur étant associer, comme l'illustre la figure 3 ci-dessous.

La table emprise et la table ouverture appartient à ce qui est appelé en archéologie, le modèle des 6 couches, ce modèle comporte en plus des tables déjà citées, les tables axe, log, poly et point. Elles correspondent aux données spatiales formant le noyau de la structure de données d'une opération archéologique lambda.

En archéologie, une emprise correspond au terrain dans lequel les archéologues peuvent effectuer leur recherche, basé sur un arrêté préfectoral. La table emprise a donc une géométrie correspondant à la surface du terrain, et d'autres champs correspondant aux informations relatives à l'intervention, comme le nom de l'opération ou encore le responsable scientifique.

Une ouverture quant à elle, correspond à tout creusement réalisé à des fins d'observations, ces ouvertures peuvent être des sondages ou des tranchées. Mais également des palier, des zones, des bermes ou des secteurs, ces derniers n'ont pas été traité pour le moment, car ce sont des cas plutôt

rares.

Comme indiqué précédemment, les tables appartenant au modèle des 6 couches correspondent aux données spatiales. Ces données sont enregistrées par le topographe, dans des couches qui lui sont propres. Parallèlement, l'archéologue renseigne les données attributaires, ici sur les tables t_tranchee et t_sondage. Une fois le travail du topographe terminé, il va alors ajouter ces données dans les couches du projet. Dans notre cas, il va simplement copier les entités de la couche ouverture sur laquelle il a travaillé, pour les coller dans la couche ouverture de BADASS. Et si la valeur correspondant au champ numéro d'ouverture de la table ouverture, est identique à celle du même champ de la table t_tranchee ou t_sondage, la géométrie sera alors identique pour les deux entités.

C'est pour réaliser cette dernière action que sont utilisées les triggers. Les triggers SQL (ou déclencheur) permettent d'exécuter un ensemble d'instruction SQL juste après un événement. Dans notre cas, nous allons par exemple, après insertion des données dans la table ouverture, et quand le type d'ouverture est 'tranchée'. Mettre à jour la table t_tranchee, en copiant la géométrie de l'élément dont la valeur du champ numéro d'ouverture de la table ouverture, est égale à celle du numéro d'ouverture de la table t_tranchee. Ce qui traduit en SQL, correspond à la figure 3, que vous trouverez en annexe.

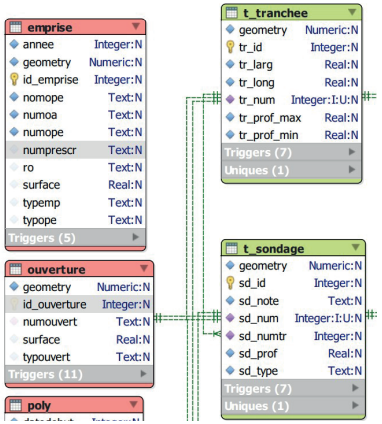


Figure 3: Schéma des tables de la base de données relative à ce stage.

4.2.3 L'identité visuelle du projet

Comme énoncé précédemment, le projet BADASS est une extension au logiciel QGIS, qui propose des formulaires afin de manipuler les données, de façon simple et intuitive, et en offrant une navigation logique entre les différents formulaires. En parallèle de la conception de la base de données, les archéologues se sont donc attelés à la tâche, afin de désigner ces formulaires pour répondre au mieux aux besoins que demande leur utilisation. Pour cela, ils ont utilisé la partie

design de l'outil Qt Creator. Qt Creator est un environnement de développement intégré (EDI ou IDE en anglais), qui fait partie du framwork Qt. Il embarque Qt Designer, un logiciel permettant la création d'interfaces graphiques, son utilisation est simple et abordable, notamment en raison du fait qu'il utilise la technologie glisser-déposer, ainsi que ces menus permettant de facilement régler les différentes propriétés. Vous trouverez un aperçu de l'interface de Qt Creator en annexe, à la figure 10. Qt Creator est orienté pour la programmation en C++*. Nous utiliserons cependant Python, avec PyQt, un module permettant d'utiliser le langage Python avec Qt.

La navigation entre les différents formulaires est synthétisée à la figure 4. Le formulaire qui apparaît au lancement de l'extension, est intitulé, le formulaire de lancement. Ce formulaire offre trois choix, qui seront détaillés par la suite. Ces choix permettent dans tous les cas d'accéder au formulaire d'accueil. Le formulaire d'accueil permet d'accéder à tous les autres formulaires, de façon directe ou indirecte. Dans notre cas le formulaire qui nous intéresse le plus est le formulaire ouverture, qui permet d'accéder aux formulaires tranchée et sondage.

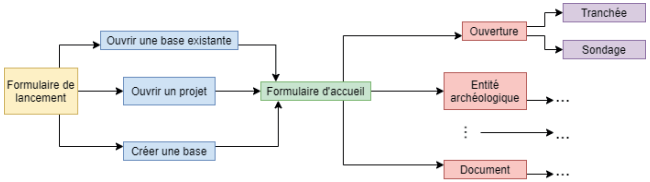


Figure 4: Diagramme de la navigation entre les formulaires

4.3 Les outils et concepts du projet : PyQGIS et PyQT

Les documents précédemment présentés m'ont été transmis lors du début de mon stage, mais avant de se lancer dans le développement de l'extension, une mise en point quant à l'utilisation QGIS était nécessaire.

4.3.1 QGIS

QGIS anciennement appelé Quantum GIS, est un logiciel SIG (Système d'Information Géographique) libre, multi-plateforme. La première version est sortie en juillet 2002. Il est écrit en C++ et en python, et utilise Qt comme support graphique. Le logiciel est grandement répandu de par sa gratuité, mais également de par les nombreux formats prise en charge. Que soit pour les couches matricielles, ainsi que les couches vectorielles. Mais aussi, car il est traduit dans de nombreuses langues. Un autre de ces points fort, est la possibilité de stocker ces données utilisées dans une base de données, avec un large choix de SGBD*, allant de PostgresSQL jusqu'à Oracle, et en passant par MySQL, DB2 et SQLite, c'est d'ailler ce dernier qui a été choisi dans ce projet comme expliqué précédemment.

Il a bien sûr des fonctionnalités communes à tous logiciels SIG, et offre de nombreux systèmes de référencement spatial. QGIS offre bien d'autres fonctionnalités qui font de lui un logiciel très utilisé par les archéologues. Il est d'ailleurs soutenu par la fondation Open Source Geospatial (OSGeo), et est recommandé par l'État français. Une des fonctionnalités qui nous intéresse plus particulièrement dans le cadre de ce projet, est celle de pouvoir réaliser des extensions. Ce qui en fait d'ailleurs une

de ces forces, puisque nous en dénombrons plus de 1400 à ce jour, répertorié et noté par les utilisateurs. Ces extensions sont réalisées en python ou en C++, à l'aide de leur API* respective, du Developer Cookbook PyQGIS et de l'aide en ligne. Dans le cadre de ce projet, il a été choisi que l'extension soit réalisée en python.

4.3.2 Les premiers rendus

Une fois le logiciel pris en main, au travers d'un tutoriel qui fait un rapide tour de vu des principales fonctionnalités. Avec une partie consacrée à python, l'environnement et l'exécution de script, l'utilisation de la console, ainsi que la création et l'utilisation d'extension et leur environnement de développement. Les premiers rendus du projet ont vu le jour, avec la création du plug-in, à l'aide de l'outil plug-in builder, comme indiqué dans le tutoriel. QGIS offre deux outils dédiés au développement d'extensions. Le premier, Plugin Builder, permet de créer les fichiers nécessaires à l'extension, et renseigne les informations préalablement demandées. Les fichiers sont écrits en python, et sont multi-plateforme. Le second est Plugin Reload, permet d'exécuter le code de l'extension. Cela permet notamment de ne pas avoir à relancer le logiciel, à chaque changement.

Lors de la création du plug-in, une erreur apparaît, elle nous indique que la ressource qui permet de compiler le fichier ressource, n'a pas été trouvé. Il nous faut donc compiler le fichier resources.qrc manuellement. Pour cela, le tutoriel nous indique la marche à suivre, il suffit de créer un fichier batch, avec le contenu indiqué, et de l'exécuter dans le répertoire du plug-in.

À la création du plug-in, deux fichiers nous intéressent plus particulièrement, il s'agit du fichier dialog_badass.py et du fichier dialog_badass_dialog.py. Le premier est très important, car il sert à initialiser le plug-in lors du démarrage de QGIS. Il comporte plusieurs méthodes, entre autres, une pour ajouter l'extension à la barre des menus de QGIS. Celle qui nous intéresse plus particulièrement est la méthode `run()`. Cette méthode est appelée à chaque lancement de l'extension, nous la modifions simplement, afin que le formulaire de lancement apparaisse au lancement. Le deuxième fichier quant à lui, correspond au fichier python qui contient la classe DialogBADASSDialog, et qui hérite du fichier ui, dialog_badass_dialog.ui. Le constructeur de cette classe va notamment utiliser la méthode `setupU()`. Après l'appelle de cette méthode, les objets créés dans le Qt Creator et contenus dans dialog_badass_dialog.ui seront maintenant accessible. Le fichier dialog_badass_dialog.py va servir de base pour la construction des fichiers python liés aux formulaires.

Après avoir construit un fichier python pour chaque fichier ui transmit au début du stage, et avoir modifié la méthode `run()` afin que d'afficher la bonne fenêtre au démarrage du plugin. Nous constatons le bon fonctionnement de l'extension. Nous voulons maintenant lier les différentes fenêtres, afin de pouvoir naviguer entre elles. Cependant, nous nous confrontons à un premier problème, comment fonctionnent les écouteurs d'événements ?

4.3.3 les écouteurs d'événements

Afin de savoir lorsqu'un événement a lieu, par exemple lorsqu'un bouton est cliqué, et effectué une action en conséquence. Nous allons utiliser le mécanisme des signaux et fentes (signals and slots).

Nous parlerons ici seulement du fonctionnement concernant PyQt5*, car il est différent pour les versions antérieures.

Un signal est quelque chose qui est généré à chaque fois qu'une action se produit, ce signal va être capté par une fente, à chaque fois qu'il est émis, si une connexion existe entre les deux. Une fente va capter le signal, et effectuer une action en conséquence, la majorité du temps la fente est une fonction ou une méthode. Le mécanisme de signal et fente présente les caractéristiques suivantes :

- Un signal peut être connecté à de nombreux emplacements.
- Un signal peut également être connecté à un autre signal.
- Les arguments de signal peuvent être de n'importe quel type Python.
- Une fente peut être connectée à de nombreux signaux.
- Les connexions peuvent être directes (c'est-à-dire synchrones) ou mises en file d'attente (c'est-à-dire asynchrones).
- Les connexions peuvent être établies entre les threads.
- Les signaux peuvent être déconnectés.

Prenons l'exemple de notre bouton, qui va nous permettre de passer à la fenêtre suivante. Nous allons donc prendre notre bouton, ici button. Et parmi les signaux qui lui sont déjà associés de par son héritage, nous allons prendre celui qui est envoyé quand le bouton est cliqué, clicked. Notre fente sera la fonction open_fenetre2, cette fonction va appeler la fenêtre 2 et va l'afficher. Il ne nous reste donc plus qu'à connecter le signal et la fente, en écrivant les choses de la façon suivante, `button.clicked.connect(open_fenetre2)`.

Nous allons aussi bien utiliser les signaux, déjà implémentés pour les objets, par exemple lorsqu'un bouton est cliqué. Lorsque qu'une Qcombobox (boîte combinée) change de valeur, et lorsque du test est mis dans un QlineEdit (champs de texte éditables). Mais aussi, ceux déjà implémentés dans l'API QGIS, par exemple quand un projet est fermé. Nous allons garder en esprit la possibilité de définir nos propres signaux, avec `pyqtSignal()` et ces différents paramètres possibles. L'utilisation de signaux créés, s'accompagne de l'utilisation de la fonction `emit()`, fonction qui sert à envoyer un signal, peut être utile également pour envoyer des signaux déjà implémenter.

Cependant, l'utilisation de signaux peut parfois poser problème, par exemple pour fermer une fenêtre à la fermeture du projet. Si la fenêtre en question n'est pas ouverte, une erreur risque d'apparaître à la fermeture du projet. Il existe de plusieurs moyens d'éviter que l'erreur ne gêne l'utilisateur. Une solution qui réglerait le problème à la racine, serait d'utiliser la fonction `disconnect()`, cette dernière permet de déconnecter un signal et une fente.

Tous les signaux se trouvent dans la documentation en ligne, et nous y trouvons plus généralement toutes les informations sur les objets, de qui il hérite, qui les hérite, leurs attributs, des fonctions, etc. Mais aussi des descriptions quant à leurs utilisations.

Après avoir compris le fonctionnement des signaux et fentes, il nous est maintenant possible de lier les différentes fenêtres entre elles et de naviguer comme la figure 4 le montre.

Parmi tous les objets utilisés dans les formulaires, une mise au point quant à l'utilisation des

QTableView est nécessaire, car leur utilisation n'est pas intuitive au premier abord.

4.3.4 Les QtableView et l'architecture modèle/vue

Les QTableView sont donc des tableaux, ces derniers utilisent l'architecture modèle/vue. L'architecture modèle/vue est une version simplifiée de l'architecture MVC, pour Model-View-Controller (ou Modèle-Vue-Contrôleur). L'architecture MVC sépare les éléments en trois parties :

- Le modèle : La partie qui contient les données.
- La vue : La partie qui s'occupe de l'affichage, elle récupère les données du modèle afin de les rendre visibles.
- Le contrôleur : La partie qui s'occupe du traitement des données.

L'architecture modèle/vue ne sépare les éléments qu'en deux parties, avec la partie modèle et la partie vue, la partie contrôleur existe toujours, elle a été intégrée dans la partie vue, afin de rendre les gestions moins complexes. Nous pouvons observer la différence entre les deux architectures, à l'aide de la figure 5.

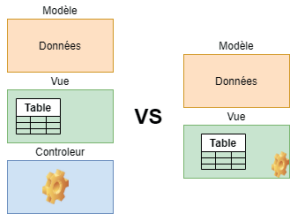


Figure 5: Architecture MVC vs Modèle/Vue

Trois objets ont le rôle de vue, QlistView (une liste d'éléments), QtreeVie(un arbre d'éléments), et la QtableView. Dans notre cas, la vue est QtableView(un tableau d'éléments), nous ne parlerons donc pas des autres vues. Pour ce qui est du modèle, il existe deux types de modèles:

- Les modèles que l'on créer soit même, en faisant hérité notre classe de QabstractItemModel, cette solution permet plus de flexibilité, mais est plus complexe à utiliser, dans notre cas le deuxième type suffira.
- Les modèles préexistants, il en existant plusieurs, avec pour chacun leur spécialité. QstringListModel (une liste de QString), QStandardItemModel (une liste d'éléments organisés sous forme d'arbre), QDirModel (la liste des fichiers et dossiers stockés sur votre ordinateur), et les modèles permettant de gérer les données issues de bases de données, QsqlQueryModel (modèle en lecture seule, basée sur les requêtes SQL), QsqlTableModel (modèle en lecture écriture, basée sur une seuls table) et QsqlRelationalTableModel (modèle identique à QSqlTableModel, avec la possibilité d'utiliser les clés étrangères)

Dans notre cas, les tableaux serviront à afficher des données, qui sont stockées dans une base de données SQLite. De plus, nous souhaitons simplement afficher les données, à l'aide de requête SQL, le modèle le plus adapté est donc QsqlQueryModel.

Maintenant que l'on a notre vue et notre modèle, il ne nous reste plus qu'à associer la vue au modèle, à l'aide de la méthode `setModel()`.

Nous allons donc utiliser QsqlQueryModel, mais faisons une mise au point quant à son utilisation. Nous savons que le modèle va récupérer les données dans la base de données, au travers d'une requête SQL. Pour cela, nous allons utiliser un objet QsqlDatabase pour la connexion avec la base de données et un objet QsqlQuery pour effectuer la requête. Pour ce qui est de QsqlDatabase, nous n'avons pas besoin de préciser le nom de la connexion, le nom de l'utilisateur et son mot de passe. Car nous utilisons une base de données SQLite, seul le fichier sqlite est à préciser.

Après avoir créé un objet QsqlDatabase, de la façon suivante, QsqlDatabase.addDatabase("QSQLITE"), la méthode `addDatabase()` sert à préciser le pilote de base de données, ici il correspond à SQLite 3. Il nous suffit d'utiliser la méthode `setDatabaseName()`, pour préciser le chemin vers le fichier sqlite, puis d'ouvrir la connexion et de la fermer au moment convenu. Pour ce qui est de QsqlQuery, les étapes sont les suivantes:

- Création d'un objet QsqlQuery
- Préparation d'une requête afin de récupérer les données
- Exécution de la requête
- Récupération du résultat de la requête
- Transmission du résultat au modèle.

La figure 5 récapitule le fonctionnement de l'architecture modèle/vue, avec un modèle qui utilise une base de données.

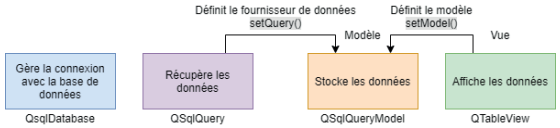


Figure 6: Diagramme récapitulatif du fonctionnement de l'architecture modèle/vue, avec le modèle QSqlQueryModel.

Afin que le tableau que nous affichons réponde à nos attendus, nous pouvons utiliser les méthodes proposées par l'API. Par exemple la méthode `setColumnHidden()`, qui sert à cacher une colonne, ou encore `setHeaderData()`, afin de renommer le nom d'une colonne.

Maintenant que les grands concepts liés à PyQt, nécessaires à la réalisation du projet sont maîtrisés, il nous reste à aborder ceux liés à PyQGIS.

4.3.5 La gestion des couches vectorielles avec PyQGIS

L'API PyQGIS est vaste, et les sujets qui peuvent être traités sont nombreux, bien heureusement ceux liés à cette partie du projet son peu nombreux. Nous allons donc aborder les sujets de la création de couche vectoriel, de l'ajout et la modification d'attributs.

L'ajout de couches vectorielles est quelque chose d'essentiel dans ce projet, et dans tout projet puisque c'est la donnée avec laquelle l'utilisateur interagit. Nous parlerons ici seulement de l'ajout d'une couche, puisque la procédure est la même pour toutes les couches. De plus, nom de la table SQL et le nom de la colonne géométrique ont été récupérés auparavant, cela sera vu plus en détail dans la suite du rapport.

Pour rappel, les couches vectorielles sont stockées dans la base de données, lors de la création d'une couche. Un accès à la base de données est donc nécessaire, cet accès se fait à l'aide d'un URI.
Un URI (Uniform Resource Identifier), est une courte chaîne de caractères identifiant une ressource sur un réseau physique ou abstrait. À noté, un URL (Uniform Resource Locator), plus familièrement appelé adresse web est d'ailleurs un sous-ensemble d'URI.

Après avoir instancié un URI, à l'aide de la classe QgsDataSourceUri, puis lui avoir défini une base de données, avec la méthode `setDatabase(path_file)`. Nous attribuons les membres liés à cette source, ses membres sont :

- Le schéma, qui est vide, simplement car SpatiaLite ne prend pas en charge les schémas
- La table
- La colonne géométrique, qui peut être null si la table n'a pas de colonne géométrique, la couche sera alors attributaire et non géométrique.

Après avoir lié ses membres à l'URI, avec la méthode `setDataSource()`. Il ne reste plus qu'à créer la couche vectorielle, en instanciant la classe QgsVectorLayer, avec en paramètres l'URI (sous forme de chaîne de caractères), le nom de la couche et 'spatialite'. Et enfin, il nous faut ajouter ces couches au projet. Pour cela, il faut récupérer l'instant du projet, et utiliser la méthode `addMapLayer()` pour ajouter la couche à la carte du projet. Les couches seront alors visibles dans la section couches de QGIS,. Pour un meilleur confort d'utilisation, il est possible de regrouper les couches, ce qui donne alors un aspect d'arbre. Cette fonctionnalité n'a pas été réalisée pour le moment, mais va l'être prochainement.

Une fois ces couches ajoutées au projet, il est par la suite possible de modifier ces attributs, de les ajouter et les supprimer. Étant donné que les couches sont stockées dans la base de données, il est donc possible de les manipuler au travers de requête SQL. Cependant, travailler de la sorte, serait se priver d'une grande partie des possibilités offertes par l'API. Une autre manière de faire, consiste à utiliser un fournisseur de données, il permet d'accéder aux entités dans la source de données, afin de les manipuler. Mais cette solution n'est pas compatible avec un point important. Pour modifier des entités dans QGIS, il est obligatoire d'être en mode édition, c'est un réflexe qu'ont tous les utilisateurs, il est donc important de conserver le même fonctionnement. Modifier des couches vectorielles à l'aide d'un tampon d'édition, vas stocker ses modifications temporairement, cela permet entre autres à l'utilisateur d'annuler les modifications, ou de les sauvegarder, et bien d'autres options. L'utilisation d'un fournisseur de données n'est pas utile lorsqu'un tampon d'édition est utilisé. Par ailleurs, PyQGIS offre la possibilité d'interagir avec le mode édition, mais nous parlerons plus en détail de cette fonctionnalité dans la suite du rapport.

En modifiant les couches vectorielles à l'aide d'un tampon d'édition, l'accès aux attributs des couches ne nécessite pas l'utilisation d'un fournisseur de données. La modification se fait donc par l'accès direct aux données. Les principales actions sont l'ajout d'entités et la modification d'attributs.

L'ajout d'une entité se fait de la façon suivante :

- Création d'une entité, instanciation d'un objet QgsFeature
- Ajout des attributs à l'entité, avec la méthode `setAttributes()`
- Ajout de l'entité à la couche, avec la méthode `addFeatures()`

La modification des attributs d'une entité se fait de la manière suivante :

- Récupération de l'entité, avec la méthode `getFeatures()` et avec une requête en paramètre
- Association de l'attribut de l'entité à une nouvelle valeur
- Mise à jour de l'entité, avec la méthode `updateFeature()`

5 Une première partie concluante

5.1 Les premiers formulaires : du lancement de l'extension à l'accueil

Une réunion a été organisée le 28 avril avec les archéologues. Cette réunion à permis de monter les premiers rendu de l'extension, mais aussi de mettre au clair leur attendu, et la procédure avec laquelle les formulaires vont être utilisés, afin que le développement de ce projet réponde au mieux à leurs attentes. En effet, les archéologues ont des procédures d'utilisation de QGIS qui leur sont propres, par exemple avec l'utilisation du mode édition. La réflexion a aussi amené des modifications dans ce qui était prévu initialement, avec par exemple la possibilité de pouvoir ouvrir un projet, et non seulement une base de données. Il a été conclu que cette partie du projet ne traite pas uniquement du formulaire d'accueil, qui est lié à la table emprise, et des formulaires ouverture.

5.1.1 Le lancement de l'extension

Avant de parler des formulaires liés à la couche emprise et aux couches ouvertures, il est nécessaire de parler du formulaire qui apparaît au lancement de l'extension. Pour information, la figure 7 représente ce formulaire.

La partie extensions concerne les réalisations d'Emma Chapuis, une autre stagiaire qui travaille au développement de l'extension. Seule l'extension Of The Dead a été commencée. Elle permet entre autres, d'ajouter un squelette, qui sert de support, pour manipuler des données. Les extensions peuvent être ajoutées à n'importe quel moment, lors de la création ou l'ouverture.

Comme le montre la figure 7, outre les possibilités d'ajouter une extension, trois choix s'offrent à nous en arrivant sur cette fenêtre. La création d'une base de données, l'ouverture d'une base de

données et l'ouverture d'un projet. Chaque choix correspond à un bouton, qui renvoie à une méthode, la connexion entre les deux se fait à l'aide des écouteurs d'événements présenter précédemment.

La création d'une base de données, est choix qui ne va être fait que très rarement, une base est créée à chaque début de projet, c'est-à-dire exceptionnellement. Lorsque qu'une nouvelle base est créée, elle doit être stockée dans un fichier sqlite, un explorateur s'ouvre, pour que l'utilisateur puisse indiquer le nom et l'emplacement de ce fichier. Une fois le chemin absolu du fichier récupéré, il est maintenant possible de créer la base est d'exécuter le code SQL fourni par les archéologues.

Cependant, deux problèmes apparaissent, le premier est dû à l'utilisation de colonne géométrique dans spatialite. Pour corriger cela, il suffit de charger l'extension mod_spatialite.dll. Le deuxième est dû à l'endroit auquel s'exécute le code python, il semblerait que ce soit à l'emplacement suivant, 'C:\Windows\System32'. Or le fichier SQL, dont nous avons besoin d'exécuter est stocké dans l'extension qui est lui-même dans le répertoire de l'utilisateur, utiliser un chemin absolu n'est donc pas possible. Heureusement, la méthode `qgisSettingsDirPath()` de l'objet QgsApplication, permet de récupérer le chemin du répertoire de l'utilisateur QGIS. Il ne reste alors, plus qu'à ajouter le reste du chemin et le nom du fichier souhaité.

Après avoir créé la base de données, il ne reste qu'à ajouter les couches au projet, puis à ouvrir la fenêtre d'accueil. La création de couche a déjà été explicitée, en partant du principe que le nom de la table et de la colonne géométrique est déjà connue. Mais nous avons bien besoin de les récupérer à un moment.

Pour cela, nous effectuons deux requêtes sur la base de données, la première permet de récupérer le nom de toutes les tables présent dans la base de données. Il faut bien sûr enlever du résultat les tables de méta données*. Après une observation sur toutes ces tables, nous remarquons qu'elles sont placées avant les tables de données et que la dernière table de méta-données est 'ElementaryGeometries', il ne reste plus qu'à garder toutes les tables contenues après cette dernière.

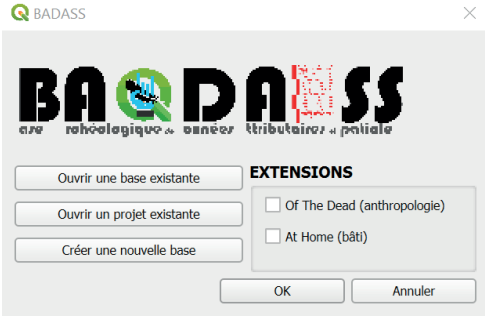


Figure 7: Formulaire qui apparaît au lancement de l'extension

Une autre partie des tables est à enlever du résultat, certaines tables ne doivent pas être visible parmi les couches du projet, étant donné que ces tables commencent par 'j_', une simple regex*

permet de s'assurer qu'elle ne sont pas présente dans les couches visibles. La deuxième requête permet de récupérer les tables contenant une colonne géométrique et le nom de leur colonne. Le tableau renvoyé pour la création des couches contient donc le nom de toutes les tables, avec le nom de leurs colonnes géométrique, si elles en ont une, ou la valeur null quand le cas contraire.

Le deuxième choix est celui de l'ouverture d'une base de données existante. Ce choix est peut commun, car les bases de données sont souvent associé à des projets, l'ouverture d'un projet est donc plus approprié. La méthode dédiée à l'ouverture d'une base de données, commence par demander à l'utilisateur au travers d'un explorateur de fichier, le fichier sqlite qui doit être ouvert. Puis reprend les trois dernière étapes de la création de la base. À savoir, récupère les tables qui doivent devenir des couches, et colonnes géométrique associer, puis la création de ces couches, et enfin l'ouverture de la fenêtre d'accueil.

Et enfin, le troisième choix, l'ouverture d'un projet, qui donc celui fait 99% du temps. Encore une fois, la personne commence par indiquer un fichier qgz (fichier contenant un projet QGIS), au travers de l'explorateur. Puis nous récupérons l'instance du projet, afin d'utiliser la méthode `read()`, pour charger le projet. Un problème se pose alors, pour récupérer le fichier sqlite associer à ce projet, afin de pouvoir se connecter à la base de données. En effet, nous pouvons imaginer plusieurs solutions afin de récupérer le chemin vers ce fichier, mais aucun ne garantit pas de fonctionner aussi bien que l'outil natif. La solution retenue récupère la couche ouverture, puis récupère le fichier dans lequel la couche est stocké, en utilisant l'URI. Évidement, cette solution ne vas pas fonctionner si l'utilisateur supprime la couche, même si la suppression de la couche ouverture à peu de chance d'arriver. Une solution potentiel, serait de d'empêcher tout utilisateur de supprimer cette couche.

Peu importe le choix qui est fait, dans tous les cas, il va nous mener à la fenêtre d'accueil. Cette fenêtre permet de gérer la couche emprise, et d'accéder au autre formulaire.

5.1.2 Le fenêtre d'accueil et le formulaire emprise

Pour rappel, la couche emprise fait partie du modèle des 6 couches. Elle est sert à décrire les informations du site, notamment la géométrie. Elle est dans un premier temps réaliser par le topographe, puis importer dans le projet. Elle est alors liés au projet, et peut être modifiée si besoin, au travers du formulaire. À l'arrivée sur cette fenêtre, les informations de la couche emprise qui peuvent être modifiés sont afficher dans le formulaire. Il est alors possible de les modifier, uniquement si la couche est mode édition, dans le cas contraire les objets sont alors désactiver.

Sur ce formulaire, et sur d'autres que nous verrons par la suite, le mode édition est présent. Afin d'éviter à l'utilisateur des aller-retour intempestif et contre productif, entre le logiciel et l'extension. Le formulaire embarque un bouton avec une image de crayon, afin de reproduire la fonction native, du passage en mode édition. Derrière cette image, se cache un mécanisme, qui permet d'avoir le bouton enfoncé lorsque le mode édition est activé, et inversement. Un bouton avec une image de disquette est également présent, elle reproduit la fonctionnalité de sauvegarde. De la même manière que la fonctionnalité native, elle est désactivée, et donc grisée, par défaut, c'est à dire lorsque le tampon d'édition est vide. Et devient active uniquement lorsque des éléments sont présent dans le

tampon d'édition.

Après que le topographe ai copié l'entité dans la couche emprise, il faut que l'affichage du formulaire se mette à jour. Pour cela, nous connectons le signal committedFeaturesAdded à la méthode qui charge les attributs dans le formulaire. Cette méthode récupère simplement les attributs puis les places dans le formulaire. Seul particularité pour l'objet Qcombobox. Si l'élément récupère dans la base de données, n'est pas présent dans les éléments initialement proposés, alors il est ajouté dans l'objet et est placé comme élément courant.

Pour modifié le tampon d'édition, cela se passe de la façon indiqué précédemment. Avec la récupération de l'entité, l'association de l'attribut de l'entité à une nouvelle valeur, et la mise à jour de l'entité. Avec l'utilisation du mode édition, il n'est pas possible d'effectuer toutes les modifications à la fermeture de la fenêtre, car l'utilisateur doit sauvegarder ces modifications avant qu'elle ne se ferme. Simplement, car le projet se ferme, lorsque la fenêtre d'accueil se ferme, nous aborderons se point plus en détail par la suite. Nous modifions donc le tampon d'édition, dès que possible. Pour cela, des signaux sont placés sur les objets, afin de savoir lorsque des modifications ont été réalisées. La fente connectée à ces signaux va simplement récupérer la nouvelle valeur. Étant donné que la méthode pour récupérer la valeur d'un objet est différentes pour tous les objets, nous utilisons la fonction `isinstance()` pour savoir le type de l'objet. Puis nous envoyons la valeur et l'attribut associer à une autre méthode, qui va effectuer la modification.

Cette autre méthode, commence par récupérer l'entité. Pour cela, elle filtre la couche en utilisant le champ id_emprise. La valeur de l'identifiant, est récupérée lors du chargement du formulaire. Cette façon de faire, implique qu'il n'est pas possible de modifier l'entité, si elle n'a pas été créée auparavant, comme c'est le cas lors de la première ouverture. Puis, les modifications ces fonts de la même manière qu'indiquer précédemment. L'attribut de l'entité est associé à la nouvelle valeur, et l'entité est mise à jour.

5.1.3 La fermeture du projet

Cette fenêtre ne contient pas seulement le formulaire de la couche emprise, elle est également la page d'accueil du projet. Elle s'ouvre quand un projet est ouvert, ou créer, il est donc normal qu'elle se ferme lorsque le projet se ferme, et inversement, que le projet se ferme lorsqu'elle se ferme.

Pour réaliser le premier point, c'est-à-dire que la fenêtre d'accueil se ferme lorsque le projet se ferme. Nous avons connecté le signal cleared, ce signal est envoyé à la fermeture d'un projet, à une fente. Cette méthode va fermer la fenêtre d'accueil, et va déconnecter le signal et fente, cela évitera les erreurs au cas où l'utilisateur essaie de fermer le projet une nouvelle fois.

Pour réaliser le deuxième point, c'est-à-dire que le projet se ferme, lorsque la fenêtre d'accueil se ferme. Nous utilisons pour cela la fonction `closeEvent(self, event)`, fonction appelée lors de la fermeture de toute fenêtre. Pour reproduire au mieux le fonctionnement de la fermeture du projet, il est important de demander à l'utilisateur, s'il souhaite enregistrer le projet, si besoin. Pour cela, après avoir regardé si le projet peut être enregistré, nous utilisons une QMessageBox, afin de proposer trois choix. L'enregistrement du projet, qui peut entraîner l'ouverture de l'explorateur, si le projet n'a jamais été enregistré, le projet est ensuite enregistré, sauf dans le cas où l'utilisateur

n'indique pas de chemin à l'explorateur et annule. Dans ce cas et dans le troisième choix, la fenêtre ne sera pas fermée. Le deuxième choix est un refus d'enregistrer le projet. La figure 8 résume les possibilités à la proposition d'enregistrement du projet, lors de la fermeture de la fenêtre d'accueil.

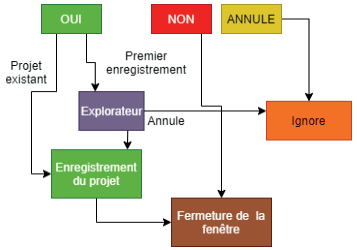


Figure 8: Diagramme des possibilités à la proposition d'enregistrement du projet, lors de la fermeture de la fenêtre d'accueil

Si les choix de l'utilisateur amènent à la fermeture de la fenêtre, cela entraîne bien évidemment la fermeture du projet. Mais aussi à la fermeture de la connexion à la base de données. Ainsi que la fermeture de toutes les autres fenêtres, si elles ne sont pas déjà fermées, cette dernière action n'a pas été implémentée pour le moment.

5.2 Les formulaires ouverture : une gestion des données qui pose les bases

5.2.1 Le formulaire ouverture

Parmi les formulaires accessibles à partir de la page d'accueil, seuls ceux liés à l'ouverture sont fonctionnels. Le formulaire ouverture ne donne pas accès aux données de la couche ouverture, mais donne accès à ceux des couches t_tranchee et t_sondage. Pour rappelle, la couche ouverture est réaliser par le topographe, qui copie les entités, puis les colle dans la couche ouverture du projet. Les triggers entrent alors en action, afin de coller la géométrie dans les couches t_tranchee et t_sondage, aux endroits où les critères sont respectés, comme indiqué précédemment.

Le formulaire ouverture affiche un tableau et une boîte combinée, cette boîte permet de filtrer par type d'ouverture, notamment sondage et tranchée. Donc chaque filtre est lié à une couche, ce qui veut dire que pour chaque filtre, le tableau change, et l'affichage lié au mode édition change également. C'est-à-dire si le crayon est enfoncé et si la disquette est disponible. Un signal a donc été placé sur la boîte combinée, afin de savoir quand elle change de valeur.

Ce signal est connecté à une fente, qui va déterminer de quelle est la nouvelle valeur et appelle la méthode appropriée en conséquence. Il existe donc une méthode pour chaque filtre, méthode qui commence par définir la couche actuelle, puis définit l'affichage des éléments d'édition, et créer un nouveau modèle pour finir par l'associé à la vue. Ce modèle est défini à partir de la classe QsqlQueryModel, comme expliqué précédemment, qui utilise une requête SQL pour récupérer les données à transmettre à la vue. Afin de rendre ce tableau plus compréhensible pour l'utilisateur, la

colonne géométrie est cachée, et les colonnes sont renommée, afin de les rendre plus explicites que les noms des colonnes de la table SQL.

Ces méthodes définissent également le nom du bouton permettant de créer une nouvelle ouverture, celui change et s'adapte, pour que l'utilisateur comprenne de quelle ouverture il s'agit. Ce bouton permet donc d'ajouter une nouvelle ouverture, le signal qui est lui est associer, est connecter à une fente. L'accès à cette méthode, et donc au formulaire qui permet saisir une ouverture se fait également au travers du tableau. En effet, il est possible d'ajouter une nouvelle ouverture, mais il est également possible de modifier une ouverture existante. Pour cela, lorsqu'une personne effectue un double-clique sur une case du tableau, une méthode permet de récupérer la valeur de l'identifiant de la ligne, ce numéro permet par la suite de filtrer la couche afin de récupérer l'entité. Puis la méthode appelle la méthode évoquée juste avant.

Cette méthode détermine le type du filtre, de la même manière que la méthode associée à la boîte combinée. Puis appelle la classe associer au formulaire correspondant, par exemple la classe DialogBADASSForm4OuvertureSondage pour le formulaire associer à la couche sondage. Cet objet prend en paramètre d'entrée la valeur de l'identifiant, s'il s'agit d'une nouvelle ouverture la valeur sera None. Ainsi qu'une valeur qui est vraie ou fausse, qui correspond à ce pourquoi le formulaire va être utilisé, la modification ou l'ajout. L'accès au formulaire n'est pas possible si la couche n'est pas en mode édition au préalable, l'utilisateur est informé si l'accès lui est refusé.

5.2.2 Les formulaires tranchée et sondage

Une fois arrivé sur le formulaire tranchée, de la même manière que pour le formulaire d'accueil avec la table emprise, les valeurs sont placées dans les objets. Cela est le cas uniquement s'il a été précisé, que le formulaire est utilisé pour une modification. Puis il est possible de modifier ou de définir les valeurs des attributs de la couche, et le bouton enregistrer permet de fermer le formulaire et d'effectuer les modifications. Une fois de retour sur le formulaire ouverture, il est alors possible d'enregistrer les modifications stockées dans le tampon d'édition.

Le fonctionnement général du formulaire permet, contrairement au formulaire d'accueil, d'effectuer toutes les modifications au même moment. Cela permet de réduire le traitement effectué à chaque fois qu'une valeur change, une seule et même modification au lieu de plusieurs. Cela permet également à l'utilisateur d'annuler les modifications qu'il est en train de faire, en utilisant le bouton annuler, ou en fermant la fenêtre.

Le choix a été fait d'utiliser un dictionnaire, afin de stocker les attributs qui ont été modifiés, et l'objet associé. Cela permet de ne pas chercher à modifier les objets qui ne l'ont pas été. Ce qui est notamment utile dans le cas où un seul objet est modifié, le processus est alors moins long. Bien que cela peut être un désavantage, par rapport à une solution qui chercherait à modifier tous les objets. Notamment, car les signaux associés à ces objets, sont, pour la plu-part envoyées à chaque fois que la valeur change, ce qui peut entraîner de multiples appelle de méthode.

Le numéro d'ouverture à deux contraintes particulière, tout d'abord le fait que la valeur doit être un entier, mais aussi qu'il doit être unique. L'objet utilisé pour saisir ce numéro est un QlineEdit, un objet qui peut recevoir n'importe quel caractère. Des vérifications sont donc nécessaires, avant

d'envoyer les modifications dans le tampon d'édition et de fermer la fenêtre.

Le bouton enregistrer et le bouton annuler, appartient à un QDialogButtonBox, et une fente est connecté par défaut pour chacun des deux boutons. Pour le bouton enregistrer, il s'agit de la fente `accept()`, c'est dans cette fente que nous avons défini le traitement nécessaire. Or, cette fente ferme la fenêtre à la fin du traitement que nous effectuons. Cela est problématique dans le cas où les informations que l'utilisateur a indiquées ne respectent pas les règles, et cause des erreurs SQL. Dans ce cas, nous indiquons à l'utilisateur de bien vouloir corriger ces erreurs, en laissant la fenêtre ouverte. Pour corriger ce problème, nous supprimons la connexion entre ce signal et cette fente, afin de définir une nouvelle connexion, qui elle ne pose pas de problème.

La méthode `accept()` commence par appeler une autre méthode, qui récupère la valeur de l'objet et effectue les traitements nécessaires si besoin. Dans le cas où les règles ne sont pas respectées, une QMessageBox est utilisée pour informer l'utilisateur de l'erreur en question. Si aucun problème n'est signalé, la valeur est ensuite stockée dans un autre dictionnaire. Et la méthode `accept()`, appelle la méthode appropriée, soit pour modifier l'entité, soit pour en ajouter une nouvelle, et enfin la fenêtre se ferme.

Les formulaires tranchée et sondage sont construits de la même manière, leur fonctionnement est identique, les seules différences sont pour les objets qu'il abrite, et pour les règles à respecter. En effet, il existe une contrainte sur le numéro de tranchée. Tout sondage appartient à une tranchée, donc la couche sondage à un numéro d'ouverture et un numéro de tranchée, pour savoir à quelle tranchée ce sondage appartient. Il existe donc une contrainte sur ce numéro, il doit obligatoirement déjà être présent dans la colonne numéro d'ouverture de la couche tranchée.

Ces deux formulaires contiennent également un tableau, une boîte combinée, et un bouton. Ces objets sont en lien avec la partie sur les entités archéologiques, que ce soit les faits, les unités stratigraphiques, etc. Cette partie n'a pas encore été réalisée, les objets n'ont pas été traités. Le tableau sert à afficher les entités archéologiques en lien avec la tranchée, par exemple. La boîte combinée sert à filtrer le tableau par type d'entité, de la même manière que le tableau d'ouverture. Elle est vide pour le moment, car les éléments qu'elle va contenir vont être probablement définis au travers d'une requête SQL. C'est d'ailler déjà le cas pour le formulaire mobilier, les éléments du filtre qu'il utilise ont été définis dans le thésaurus*, et sont récupérer à l'aide d'une requête SQL. Notre thésaurus est défini dans la base de données, et contient plusieurs listes, comme une liste de matières, ou une liste de couleur, qui peuvent être utilisées pour remplir des boîtes combinées.

5.3 Bilan de la première partie et suite du projet

5.3.1 Une première partie de projet concluante

Tous les éléments qui viennent s'être présentés constituent une première version de l'extension. Cette version nécessite quelques améliorations pour la partie emprise et ouverture correspondent pleinement aux attentes. Mais la version est utilisable en l'état, bien qu'elle ne le sera sûrement pas, car ce n'est qu'une partie du projet.

Au cours du projet, les archéologues ont pu tester les différentes améliorations, quand une version était disponible, sur le gitlab* qu'ils utilisent dans le cadre du projet. Ils ont alors pu faire des retours, sur ce les choses à améliorer, de nouvelles idées, sur ce qu'ils pensaient. Au cours de la dernière réunion, la dernière version, qui est la version actuelle, leur été présentée avec une démonstration. Les avis étaient alors positifs sur le travail accompli.

Un point ne peut néanmoins être solutionné, lors de la création d'une nouvelle base de données, la fenêtre d'accueil met du temps à apparaître. Il arrive même que l'application semble ne plus fonctionner, avant que la fenêtre apparaisse. Des tests ont été réalisés pour trouver l'origine du problème, et une solution si possible. En utilisant la fonction `time()` du module time, il est possible d'obtenir le temps d'exécution d'un morceau de code. Le temps observé pour l'exécution, de la méthode qui créer la base de données, est d'environ cinq secondes et 5 centièmes de seconde.

Cette méthode comprend, la création de la base de données et la connexion à la base de données. Puis la récupération des tables et colonnes géométrique, qui est utile pour la création des couches. Et enfin l'ouverture de la fenêtre d'accueil. Le tableau suivant est le résultat des tests effectué, en ayant observé le temps entre les actions.

Actions de la méthode	Temps (en secondes)	Temps (en pourcentage)
Création de la base de données	4	72
Connexion à la base de données	0,02	0
Récupération du tableau	0,03	0
Création des couches	1,4	25
Ouverture de la fenêtre d'accueil	0,09	1

Nous remarquons donc au travers de ce tableau que ce qui prend le plus de temps est la création de la base de données. Cette action se fait au travers de la fonction `create_bdd()`. Cette fonction commence par se connecter à la base de données, ce qui la crée au passage. Puis, charge l'extension comme indiqué précédemment, et initialise les données spatiales. Et enfin, lit le fichier SQL et exécute le code, après avoir récupéré le nom du fichier. Les tests sont effectués pour voir ce qui prend le plus de temps dans cette fonction, sont regroupés dans le tableau suivant.

Actions de la méthode	Temps (en secondes)	Temps (en pourcentage)
Connexion à la base de données	0,002	0
Chargement de l'extension	0,05	0
Initialisation des données spatiale	0,3	5
Récupération du chemin du fichier	0,05	0
Exécution du fichier SQL	5,3	95

Bien évidemment, il faut prendre en compte, les conditions dans lesquelles ont été effectué ces tests. Ces tests ont été réalisés sur un ordinateur portable, et ont été relevés de manière approximative, de façon à avoir un aperçu des résultats.

Nous remarquons une première chose étrange, le temps total est supérieur de plus d'une seconde, et

cinq centième, à celui mesurer pour la création de la base de données. Le fait de sonder le temps n'explique pas ce résultat anormal.

Le résultat qui nous intéresse est celui du temps pour l'exécution du fichier SQL. C'est donc la principale cause, de la perte de temps, lorsque nous choisissons de créer une base de données. Cela est dû à la taille du fichier SQL, qui est de 54 kilo-octets. Malheureusement, il n'est donc pas possible de régler ce problème.

Nous notons également que l'initialisation des données spatiales se fait en utilisant la fonction `InitSpatialMetaData()`, dans une requête SQL. Le fait de passer la valeur 1 en paramètre réduit considérablement le temps d'exécution de la fonction. Le temps est réduit, car l'ensemble de l'opération est traité comme une seule transaction, ce qui est plus rapide, mais moins sûr.

5.3.2 Une projet plein d'avenir

La suite du projet va être réalisée par Emma Chapuis dans un premier temps, puis par Jean-Baptiste Barreau et par d'autres étudiants. Les prochaines fonctionnalités qui vont être implémentées, concernent les formulaires entités archéologiques. La façon dont ces formulaires vont être construits, est identique à celle des formulaires ouvertures. De la même façon, le premier formulaire va contenir un tableau avec un filtre et une boîte combinée, ainsi qu'un crayon et une disquette. Puis les formulaires suivants vont contenir des objets, qui vont être utilisés de la même manière.

Il reste cependant quelques améliorations à faire concernant ce qui a été réalisé, certaines ont déjà été énoncées dans le rapport. Par exemple, séparer les différentes couches dans des groupes, afin que l'utilisateur ne soit pas perdu dans ce qu'il fait. Ou encore, trouver une meilleure solution, pour récupérer la base de données associée au projet, lors de son ouverture. À défaut, empêcher l'utilisateur de supprimer la couche emprise, bien que cela ne sera valable que dans le cadre de l'utilisation de l'extension. Il est également intéressant, de trouver une manière d'intégrer aux formulaires, la fonctionnalité de collage des entités, afin d'améliorer l'expérience utilisateur. Ainsi que de trouver une procédure permettant d'utiliser les images directement dans Qt Creator. Pour l'instant, cela est possible, mais l'image n'apparaît pas sur le formulaire lors de son utilisation, afin de palmier cela, l'image est ajoutée en python.

Un point qui n'a pas été abordé dans les parties précédentes du rapport, car non-fonctionnel à ce jour. Ce point est le fait de pouvoir sélectionner sur QGIS, une entité lorsqu'elle est sélectionnée sur un tableau. Cette fonctionnalité n'est pas fonctionnelle, car le filtre qu'il produit ne fonctionne pas, que ce soit avec l'extension, mais aussi avec les outils natifs. Malgré le fait que ce filtre respecte les normes d'écriture. De plus, il a été observé que le filtre fonctionne avec une seule entité.

6 Conclusion

Le stage que j'ai effectué m'a permis de développer mes compétences techniques, que ce soit en python, langage que j'ai déjà utilisé, dans le module qui lui est consacré, ainsi que pendant les projets tutorés auxquels j'ai participé. Mais également avec l'utilisation des interfaces graphiques, que j'ai pu apprendre au cours de mon DUT. Les principaux concepts découverts en Java au travers des bibliothèques Swing et AWT s'appliquent également avec PyQt, bien que nous avions utilisé un autre mécanisme pour les écouteurs d'événements.

J'ai pu développer des compétences plus générales, relatives à la programmation, sur la manière de se servir des documentations en ligne, et sur le partage des ressources avec la découverte de gitlab. Mais aussi relative à la gestion d'un projet et à la vie en entreprise, bien ce stage c'est tenue en télétravail. Cela m'a permis de communiquer avec les clients pour comprendre les attentes, d'avoir des échanges sur l'avancé du projet, sur des points à éclaircir.

Le stage a commencé avec une prise en main de l'environnement du projet. Afin de bien comprendre le contexte du projet, mais aussi apprendre à maîtriser les différents concepts. Que ce soit l'utilisation de PyQt, mais également du logiciel QGIS dans un premier temps, puis la conception de l'extension et de son environnement de développement. Puis de son API PyQGIS, apprendre à utiliser les outils utiles pour la conception du projet. Une fois les principaux concepts maîtrisés, la conception a alors débuté. Le développement c'est procédé par étapes, avec pour commencer le formulaire de lancement, puis celui d'accueil. Pour terminer par les formulaires d'ouverture, qui posent un modèle qui va être repris pour les autres types de données. Pour chaque formulaire, des tests ont été effectués pour être sûrs de ne pas provoquer d'erreur. Chaque nouvelle version était envoyée aux clients, qui effectuaient alors un retour sur les choses à améliorer et des corrections à apporter.

Le projet va continuer à évoluer, il va être repris successivement par différentes personnes afin de le mener à son terme. Et peut-être par un stagiaire de l'IUT. Les technologies que ce projet utilise ont encore de beaux jours devant elles, que ce soit le langage python. Mais également le logiciel QGIS, avec une communauté active. Une fois terminé, il servira de référence pour beaucoup d'archéologues, de par les fonctionnalités qu'il propose.

L'expérience acquise lors de ce stage va m'être utile dans mes prochaines expériences, aussi bien professionnelle, que personnelle.

7 Bibliographie

Documentation Developper cookbook : https://docs.qgis.org/3.16/en/docs/pyqgis_developer_cookbook/index.html

Documentation PyQGIS : https://qgis.org/pyqgis/master/

NEBRA Mathieu, Découvrez l''architecture MVC avec les widgets complexes :

https://openclassrooms.com/fr/courses/1894236-programmez-avec-le-langage-c/1902176-decouvrez-l-architecture-mvc-avec-les-widgets-complexes

Documentation PyQt5 : https://doc.qt.io/qt-5/

8 Glossaire

API : Application Programming Interface ou Interface de Programmation d'Applications - ensemble normalisé de classes, de méthodes, de fonctions et de constantes qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels.

Archéosciences : Science de l'archéologie

BADASS : Base Archéologique de Données Attributaires et SpatialeS

BDD : Base De Données - permet de stocker et de retrouver des données structurées, semi-structurées ou des données brutes ou de l'information.

CNRS : Centre national de la recherche scientifique - plus grand organisme public français de recherche scientifique.

CReAAH : Centre de Recherche en Archéologie, Archéosciences, Histoire

C++ : Langage de programmation.

Fouille : l'acte de rechercher des vestiges enfouis, qu'il s'agisse de constructions, d'objets ou de traces de l'activité humaine passée, et de procéder à leur mise au jour par enlèvement des matériaux et sédiments qui les recouvrent.

Inrap: Institut national de recherches archéologiques préventives - établissement public à caractère administratif de recherche français

Méta-données : Donnée servant à définir ou décrire une autre donnée

GitLab : Logiciel libre de forge basé sur git proposant les fonctionnalités de wiki, un système de suivi des bugs, l'intégration continue et la livraison continue

QGIS : Logiciel SIG libre multiplate-forme

Paléoenvironnementalistes : Scientifique spécialisé dans l'étude des environnements anciens.

PyQt5 : Version cinq de PyQt, un module libre qui permet de lier le langage Python avec la bibliothèque Qt.

Python : Langage de programmation.

Regex : Expression régulière - chaîne de caractères, qui décrit, selon une syntaxe précise, un ensemble de chaînes de caractères possibles.

SIG : Système d'Information Géographique - système d'information conçu pour recueillir, stocker, traiter, analyser, gérer et présenter tous les types de données spatiales et géographiques.

SGBD : Système de Gestion de Base de Données - logiciel système servant à stocker, à manipuler ou gérer, et à partager des données dans une base de données.

SQL : Structured Query Language - langage informatique normalisé servant à exploiter des bases de données relationnelles. en garantissant la qualité, la pérennité et la confidentialité des

informations, tout en cachant la complexité des opérations.

Thésaurus : Liste organisée de termes contrôlés et normalisés

Triggers : Déclencheur - en programmation procédurale, un déclencheur est un dispositif logiciel qui provoque un traitement particulier en fonction d'événements prédéfinis.

Unités stratigraphiques : Découpages proposés pour définir des séquences de dépôt.

UPR : Unités Propres de Recherche – entité exclusivement gérée par le CNRS.

URM : Unité Mixte de Recherche - entité administrative créée par la signature d'un contrat d'association d'un ou de plusieurs laboratoires de recherche d'un établissement d'enseignement supérieur (notamment d'université) ou d'un organisme de recherche avec le CNRS.

9 Annexes

```
CREATE TRIGGER trgai_ouverture_maj_t_tranchee
    AFTER INSERT
    ON ouverture
FOR EACH ROW
WHEN (NEW.typouvert = 'tranchée')
   BEGIN
   UPDATE t_tranchee
   SET geometry = NEW.geometry
   WHERE NEW.numouvert = tr_num ;
END ;
```

Figure 9: Trigger SQL utilisé pour la copie de la géométrie

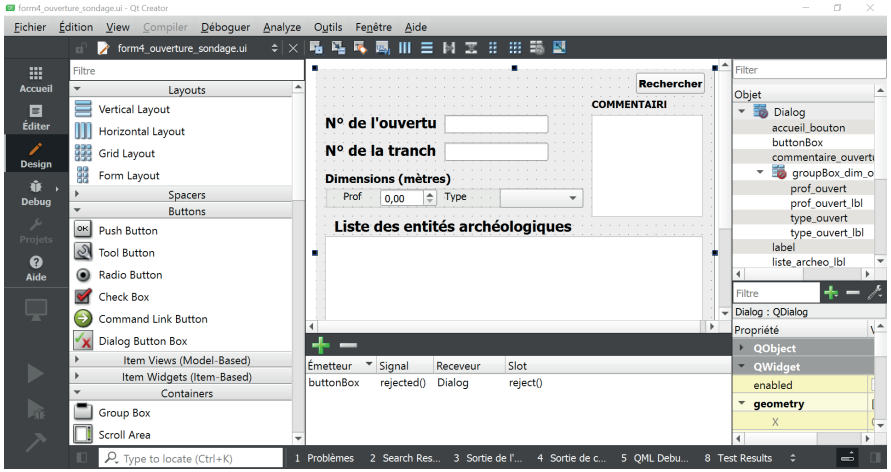


Figure 10: Aperçu de l'interface de Qt Creator

Table des matières