CERTIFICAT COMPLÉMENTAIRE EN GÉOMATIQUE

Création automatisée de fiches d'identités pour les piézomètres du canton de Genève à l'aide d'outils de géo-programmation

Présenté par:

Giulia Marti

Université de Genève

Réalisé dans le cadre d'un stage au Service de géologie, des sols et des déchets (GESDEC) de la République et du Canton de Genève

Supervisé par Dre. Stéphanie Favre et Prof. Anthony Lehmann

Janvier 2022

UNIVERSITÉ DE GENÈVE
FACULTÉ DES SCIENCES DE LA SOCIÉTÉ

REPUBLIQUE ET CANTON DE GENEVE
POST TENEBRAS LUX

RÉSUMÉ

L'objectif de ce travail consiste en la création automatisée de fiches d'identités pour les équipements de piézomètres du canton de Genève. Ce travail a été réalisé dans le cadre d'un stage de quatre mois au Service de géologie, des sols et des déchets de l'Etat de Genève (GESDEC). Un réseau de piézomètres, dispersés à l'échelle du canton et avec une multitude de différentes caractéristiques techniques, permet au GESDEC d'assurer le monitoring des eaux souterraines. Les géoinformations liées aux piézomètres sont répertoriées dans le système d'information du sous-sol (SOLSTISS). Ce travail propose une méthodologie qui permet d'automatiquement créer et actualiser des fiches d'identités qui contiennent un résumé des caractéristiques clés pour chaque piézomètre, y compris des images, à travers un script Python. Les fiches d'identités sont mises à disposition en format PDF sur un serveur accessible à l'ensemble du GESDEC. La plus grande limitation à l'heure actuelle reste le temps d'exécution pour la totalité du script. En conclusion, la création de ces fiches d'identité de piézomètres automatisée permet donc d'optimiser la mise à disposition de géoinformations liés aux piézomètres, avec des données actualisées et dans un format compatible avec les différents besoins du service.

REMERCIEMENTS

En premier lieu, je tiens à remercier le GESDEC de m'avoir donné l'opportunité de réaliser mon stage au sein de leur service et de vivre une expérience professionnelle enrichissante.

En particulier, je souhaite adresser mes remerciements à Dre. Stéphanie Favre, pour son encadrement, ses conseils, son écoute pendant toute la durée du stage, qui m'ont permis de de découvrir le monde des géodonnées (hydro)géologiques et de mener à terme ce projet de stage. De plus, je remercie également tous les collègues du service pour l'accueil chaleureux qui m'a été réservé ainsi que les retours constructifs tout au long de mon stage. Je souhaite également remercier Prof. Anthony Lehmann pour le suivi académique et ses conseils, qui ont permis de relier les exigences académiques et celles du service.

Je tiens à exprimer ma reconnaissance à toutes les personnes impliquées dans le Certificat complémentaire en géomatique, en particulier les enseignants, pour tout le savoir et les compétences transmises dans le cadre de cette formation.

Et puis un grand merci à ma famille et mes ami·e·s, pour leur présence et soutien qui a su m’encourager pendant ces derniers mois.

TABLE DES MATIÈRES

Table des figures

Table des tableaux

Table des scripts

1. INTRODUCTION

Les défis liés aux eaux souterraines sont nombreux, tel que la surexploitation ou bien la pollution des ressources en eau (OFEV, 2020). Cependant elles sont d'une grande importance pour maintenir le système hydrique local en équilibre et représentent une importante source d'eau potable. Par conséquent, la supervision de l'état et du niveau de ces nappes est primordiale. Dans le canton de Genève, le Service de géologie, des sols et des déchets (GESDEC) est responsable de la surveillance de ces eaux souterraines. Le GESDEC effectue des mesures régulières sur les nappes du canton et répertorie également les piézomètres réalisés par des entreprises privées. Ces données sont enregistrées dans la vaste base de données du service. Ces informations clés servent comme base de renseignement pour les professionnels qui souhaitent utiliser ces équipements et par exemple effectuer des relevés sur les piézomètres.

L'objectif de ce projet de stage est la création de fiches d'identités qui permet de répertorier les informations relatives à chaque piézomètre sous forme d'un résumé. Ceci permettrait une meilleure valorisation des données géospatiales et attributaires déjà existantes et faciliterait leur utilisation, permettant d'accéder à toutes les informations simultanément.

Ce travail présente le processus de travail pour la création des fiches d'identités en cinq parties principales. Dans une première partie, le contexte du travail est développé, notamment les enjeux liés aux eaux souterraines, le rôle du GESDEC en lien avec ces ressources ainsi que l'objectif du stage. Dans la partie méthodologique, les caractéristiques principales de la fiche, les catégories de données utilisées ainsi que le type de logiciel sont introduits avant de poursuivre avec les détails liés au prétraitement des données et la création des fiches à travers des outils de géo-programmation. Dans la partie résultats, une sélection de fiches d’identité générées est présentée. La partie discussion permet de formuler des réflexions critiques sur le processus et le résultat en soi ainsi que le déroulement du stage.

2. CONTEXTE

LES EAUX SOUTERRAINES

Les eaux souterraines désignent la présence d'eau dans le sous-sol, qui se localise et circule dans les sols meubles ou bien à travers les interstices des roches (OFEV, 2019). La recharge se réalise à travers l'infiltration d'eau de précipitation dans le sol ou bien depuis les rivières et lacs (cf. Figure 1) (OFEV, 2019).

En Suisse, les aquifères représentent le deuxième plus important réservoir d'eau en Suisse (après les lacs) et fournissent 80% de l'alimentation en eau potable (OFEV, 2019 ; 2020). Les sédiments du sol permettent de « filtrer » l'eau, ce qui a un effet bénéfique sur la qualité de l'eau des aquifères, notamment en termes de potabilité. Les eaux souterraines jouent également un important rôle pour le cycle hydrologique, en alimentant les sources, rivières et lacs et contribuent à préserver les habitats aquatiques cruciaux pour la biodiversité. De plus, dans le contexte de la transition énergétique, les capacités thermiques des nappes phréatiques sont de plus en plus utilisées et exploitées par la géothermie (OFEV, 2020).

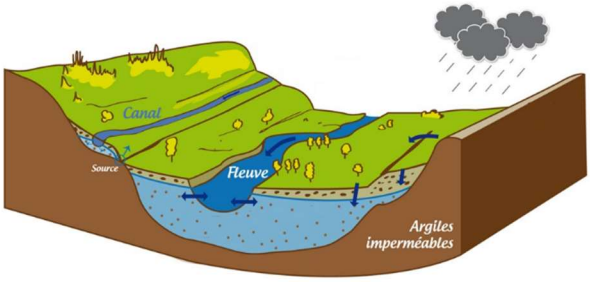


Figure 1: Représentation graphique des eaux souterraines d'une nappe alluviale. La nappe est alimentée par les précipitations, les ruissellements ainsi que par les cours d'eau. L'alimentation entre le fleuve et la nappe est réciproque et dépendante du niveau d'eau du fleuve et de la nappe. (Source de l'image : Syndicat Mixte pour la protection et la gestion des nappes souterraines de la plaine du Roussillon (s.d.), adapté par l'auteure.)

Cependant, les pressions sur ces ressources augmentent, notamment à travers la présence de substances de synthèse, tel que des produits phytosanitaires et des polluants émanant de différentes activités anthropiques (OFEV, 2020). Ces substances peuvent causer une dégradation importante de la qualité des eaux de surface et souterraines (République et Canton de Genève, s.d.-a), avec des conséquences environnementales importantes. De plus, le changement climatique, avec une altération du cycle hydrologique, impacte également les nappes phréatiques, notamment lors de sécheresses (OFEV, 2019). Le monitoring et une gestion et protection adéquate de ces ressources est donc primordiale afin d'assurer la pérennité des eaux souterraines et de leurs fonctions pour l'environnement et la société.

PRÉSENTATION DES NAPPES GENEVOISES

A Genève, quatre nappes principales peuvent être distinguées : La nappe de l'Allondon, la nappe du Genevois, la nappe de Montfleury et la nappe du Rhône (cf. Figure 2). La caractéristique de « nappe principale » est déterminée en fonction de la capacité de la nappe. Pour une nappe principale, le débit requis est fixé à 300 litres par minute, la surface doit être importante et la nappe doit se situer à une profondeur entre 15 et 60 mètres depuis la surface (République et Canton de Genève, s.d.-a). En plus des nappes principales, les nappes superficielles et temporaires complètent la carte des aquifères dans le canton de Genève. Les aquifères se situent dans de la roche meuble, qui induit généralement une importante perméabilité des sols (OFEV, 2019 ; République et Canton de Genève, s.d.-a).

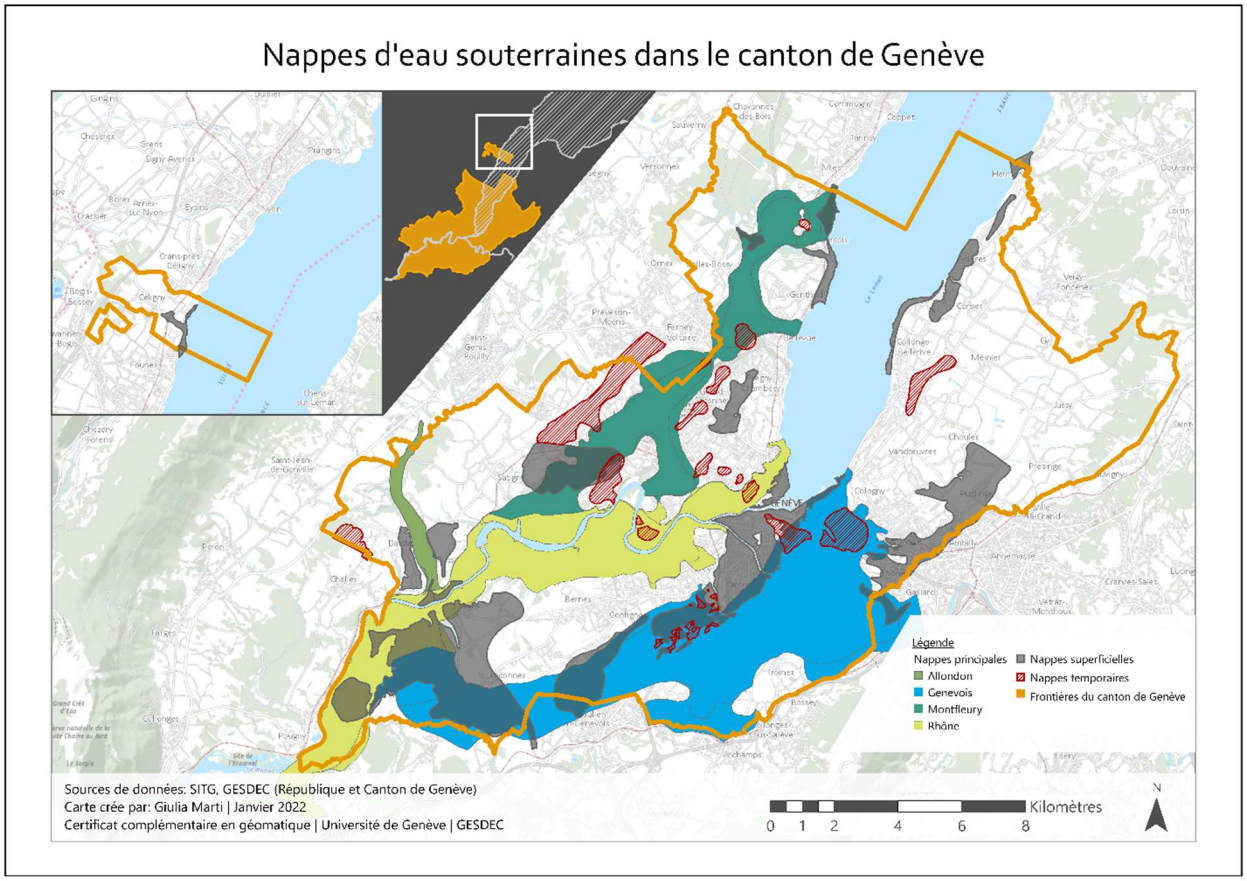


Figure 2: Représentation graphique des aquifères présents sur le territoire du canton de Genève. Les nappes principales sont celles de l'Allondon, du Genevois, de Montfleury et celle du Rhône. Elles sont visualisées par des couleurs dans le spectre vert-bleu. Les nappes superficielles sont illustrées en gris et les nappes temporaires en rouge. (Source : Créé par l'auteure).

Les nappes souterraines fournissent entre 10 et 20% de l'eau potable sur le canton, le reste de l'alimentation est assurée par l'eau du lac (République et Canton de Genève, s.d.-a). Actuellement, ce sont uniquement la nappe de l'Allondon et celle du Genevois qui sont exploitées pour l'alimentation en eau potable (République et Canton de Genève, s.d.-a).

La nappe du Genevois constitue le principal réservoir d'eau souterraine dans le canton de Genève avec 80 millions de mètres cubes d'eau et s'étend jusqu'en en France voisine, en Haute-Savoie (République et Canton de Genève, s.d.-a). La recharge de la nappe se réalise à travers les précipitations et en cas de crue de l'Arve (République et Canton de Genève, s.d.-a). L'importance de cette nappe pour l'alimentation en eau potable demande donc un suivi précis et très régulier des niveaux d'eau de la nappe.

La nappe de l'Allondon est également exploitée pour le captage d'eau potable. Les faibles débits de l'Allondon, surtout en été, peuvent cependant rendre délicat l'utilisation de cette nappe (République et Canton de Genève, s.d.-a). Le programme GEothermie 2020, explorant le potentiel de la géothermie pour réussir la transition énergétique du canton, inclut également une analyse des caractéristiques des nappes de Montfleury et du Rhône, ce qui permettrait de déterminer le potentiel d'utilisation de ces eaux souterraines (République et Canton de Genève, s.d.-a).

Les nappes souterraines du canton sont suivies par le Service de Géologie, des Sols et des Déchets (GESDEC), qui effectue régulièrement des mesures du niveau d'eau sur les différentes nappes. Le service des sites pollués effectue également des mesures pour contrôler la qualité des eaux souterraines.

PIÉZOMÈTRES

Le niveau d'eau des nappes souterraines est mesuré à partir de piézomètres. Un piézomètre est un équipement au sein d'un sondage réalisé depuis la surface, qui dispose d'une partie perméable (crépine) qui permet l'infiltration d'eau dans le tube (cf. figure 3). En mesurant la distance depuis la surface du terrain jusqu'à la surface de l'eau dans le piézomètre et en utilisant l'altitude du terrain comme base de calcul, le niveau d'eau de la nappe peut être déterminé. Les piézomètres permettent donc le suivi des niveaux d'eau des nappes et de déterminer les fluctuations. Les piézomètres sont également réalisés dans le cas de travaux de construction, permettant de déterminer le niveau de la nappe à cet emplacement précis et d'adapter le projet en fonction.

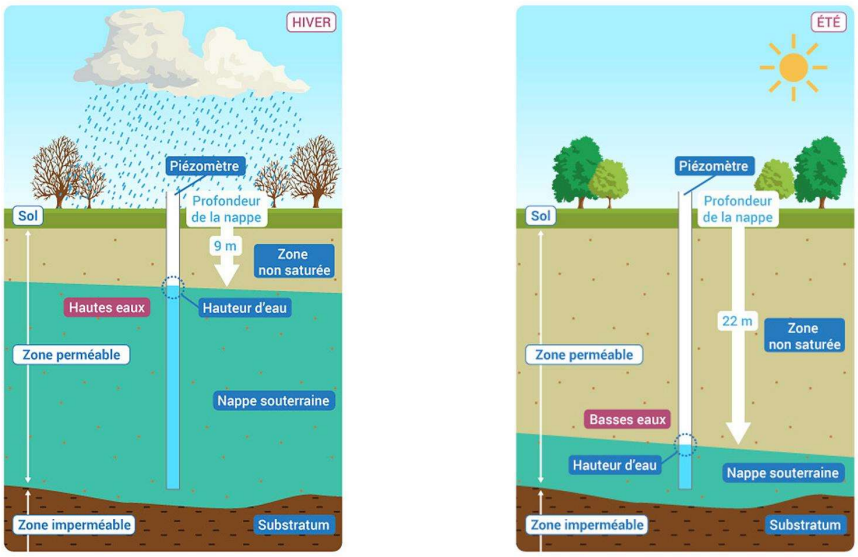


Figure 3: Schéma expliquant le fonctionnement d'un piézomètre. Le niveau d'eau de la nappe peut être déterminé en mesurant la distance entre la surface et le niveau d'eau de nappe déterminé dans le piézomètre (9m dans le schéma de gauche et 22m dans celui de droite). Dans le schéma de gauche, le niveau de nappe est donc plus haut que dans le schéma de droite. A travers les piézomètres, les différents niveaux d'eaux des aquifères peuvent ainsi être mesurés et suivis afin de détecter les fluctuations en fonction des saisons et suite à des évènements hydrologiques. (Source de l'image : Eaufrance (2019)).

Le GESDEC recense tous les piézomètres réalisés sur le canton de Genève. Plus de 5000 piézomètres sont inscrits dans la base de données. La figure 4 montre la répartition géographique des piézomètres sur le territoire. Il reste à préciser que pas tous les piézomètres illustrés dans la figure existent encore à l'heure actuelle. De nombreux piézomètres sont uniquement utilisés pour effectuer une mesure et détruits lors des travaux de construction subséquents. D'autres peuvent être inactifs ou bien défectueux. Les informations techniques sur les piézomètres installés lors de travaux de constructions sont transmises au GESDEC par les bureaux de géologie responsables du forage. Environ 550 piézomètres ont été réalisés sur mandat du GESDEC ou bien ont été intégrés aux piézomètres du GESDEC après la fin de travaux de constructions. Ces derniers sont régulièrement utilisés et servent à assurer le suivi des nappes phréatiques du canton.

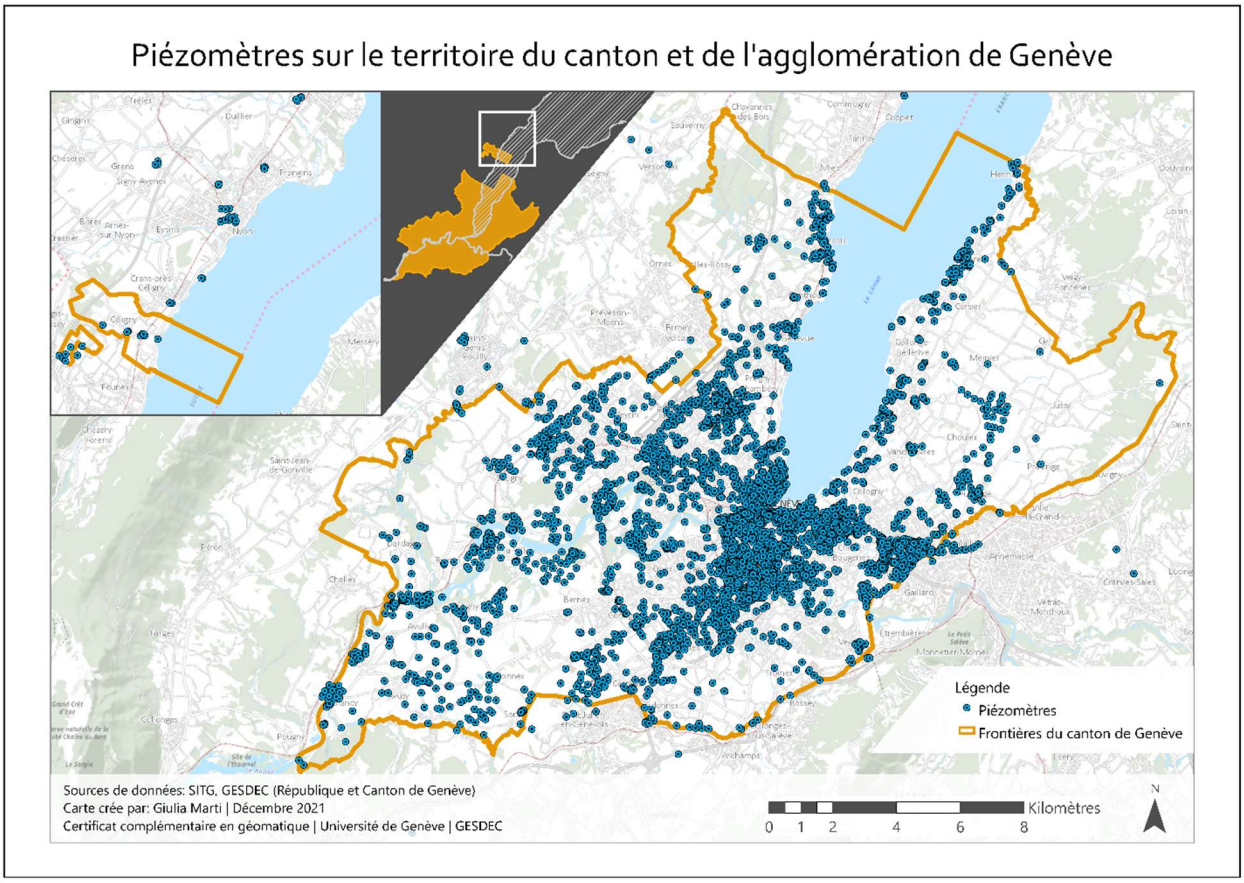


Figure 4: Représentation graphique des piézomètres recensés sur le canton de Genève et son agglomération. L'état des installations peut être actif, inconnu, défectueux ou bien détruit. La carte représente ainsi un résumé de tous les équipements de piézomètres réalisé au cours des années. (Source : Créé par l'auteure).

SERVICE DE GÉOLOGIE, DES SOLS ET DES DÉCHETS

PRÉSENTATION DE L'ORGANISATION HÔTE

Le GESDEC est le service de géologie, sols et déchets, rattaché à l'Office de l'environnement de la République et du canton de Genève et fait partie du département du territoire (DT-OCEV-GESDEC). Les activités du GESDEC concernent différents aspects de l'environnement liés aux domaines du sous-sol, des sites pollués et des déchets.

Au sein du GESDEC, le stage de géomatique s'est effectué au sein du secteur des sols et sous-sols. Ce dernier a pour objectif de *"protéger, gérer et exploiter durablement les sols, le sous-sol et les eaux-souterraines*" (République et Canton de Genève, 2019, p. 2). Ces ressources relèvent d'une grande importance pour la transition écologique du canton, notamment à travers le potentiel géothermique. Ainsi, le Plan de Gestion des ressources du

sous-sol a été élaboré, distinguant trois objectifs prioritaires (République et Canton de Genève, 2020a) :

1. *Coordonner les développements*
2. *Acquérir et diffuser de nouvelles connaissances*
3. *Clarifier les conditions d'exploitation*

Les nappes souterraines sont parties intégrantes de ce Plan de Gestion des ressources du sous-sol. La gestion de données géologiques sur les sols et les eaux souterraines joue un rôle clé pour fournir les informations liées à ces ressources et fait partie des responsabilités du secteur.

Tel que mentionné auparavant, le GESDEC est responsable du suivi des nappes souterraines. Ceci comprend la mesure des niveaux d'eau sur le terrain, l'insertion de ces informations dans le système puis le traitement et l'analyse de ces données, permettant de déterminer les évolutions et l'état des ressources souterraines.

SOLSTISS : SYSTÈME D'INFORMATION DU SOUS-SOL

Au cœur de la gestion de ces données au GESDEC se trouve le système d'information SOLSTISS (SOL-Système Territorial d'Information du Sous-Sol). Le développement de SOLSTISS s'inscrit dans l'objectif du Plan de Gestion des Ressources d'améliorer les connaissances du sous-sol cantonal et d'avoir des informations fiables sur lesquelles appuyer les décisions de gestion des ressources (République et Canton de Genève, 2020a). La figure 5 visualise les différentes fonctions de SOLSTISS qui servent en tant que support à la prise de décision. SOLSTISS sert à *« systématiquement intégrer, conserver, traiter et administrer tous les résultats issus des travaux de prospection, exploration et exploitation effectuées dans le sous-sol »* (République et Canton de Genève, s.d.-b). Ce système d'information est constamment enrichi avec les nouvelles informations émanant de projets réalisés à travers le canton. Ce caractère dynamique permet donc de mettre à disposition des données actualisées et d'améliorer les cartes et modèles, puis de les diffuser aux professionnels du domaine et au grand public (République et Canton de Genève, s.d.-b).

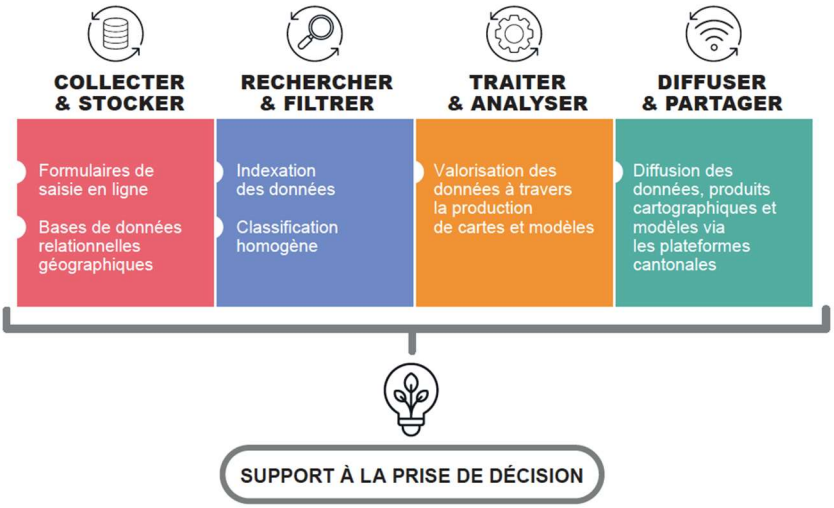


Figure 5: Fonctions du système d'information du sous-sol genevois (SOLSTISS). (Source: République et Canton de Genève, 2020a).

Le travail de ce stage s'est focalisé sur la base de données relationnelle géographique de SOLSTISS, qui fait partie du pilier fonctionnel « Collecter et stocker ». Cette base de données fournit notamment des informations géospatiales, telles que l'emplacement d'un sondage ou bien d'un équipement, ainsi que des informations liées dans des tables attributaires. Ce sont ces informations qui ont été au cœur de ce projet de stage.

OBJECTIF DE RECHERCHE/DU STAGE

L'objectif de ce projet de stage est la création de fiches d'identité pour tous les piézomètres répertoriés dans la base de données relationnelle de SOLSTISS. Cette fiche d'identité a pour but de mettre à disposition un résumé avec toutes les informations disponibles pour chaque piézomètre, dans un format utilisable au bureau ainsi que sur le terrain. Actuellement, les informations sont consultables dans la base de données SOLSTISS, mais se trouvent dans de nombreuses tables liées qui doivent être ouvertes individuellement dans ArcGIS pour extraire l'information souhaitée. L'objectif du travail est donc de regrouper ces informations dans un format pratique afin de pouvoir les consulter simultanément. Il est cependant important que les données soient régulièrement mises à jour, au vu de la caractéristique dynamique de SOLSTISS. Cette fiche d'identité serait destinée aux collaborateurs·trices du GESDEC et aux partenaires, qui auraient besoin d'informations générales sur un piézomètre ou bien devraient se rendre sur le terrain et y effectuer une mesure.

2. MÉTHODOLOGIE

La partie méthodologique introduit dans un premier temps les caractéristiques principales de la fiche d'identité ainsi que les catégories de données et logiciels utilisés dans le cadre de ce travail. Ensuite, le prétraitement des données est présenté avant de passer en revue les différentes étapes du script Python permettant de créer les fiches d'identité de piézomètre.

DÉTERMINATION DES CARACTÉRISTIQUES PRINCIPALES

Le cœur de ce projet est la valorisation des données existantes. Cette valorisation consiste en un regroupement de l'information et une mise à disposition dans un format convenable aux besoins du métier. Dans un but premier, les fiches d'identité doivent comprendre un résumé des caractéristiques individuelles de chaque piézomètre.

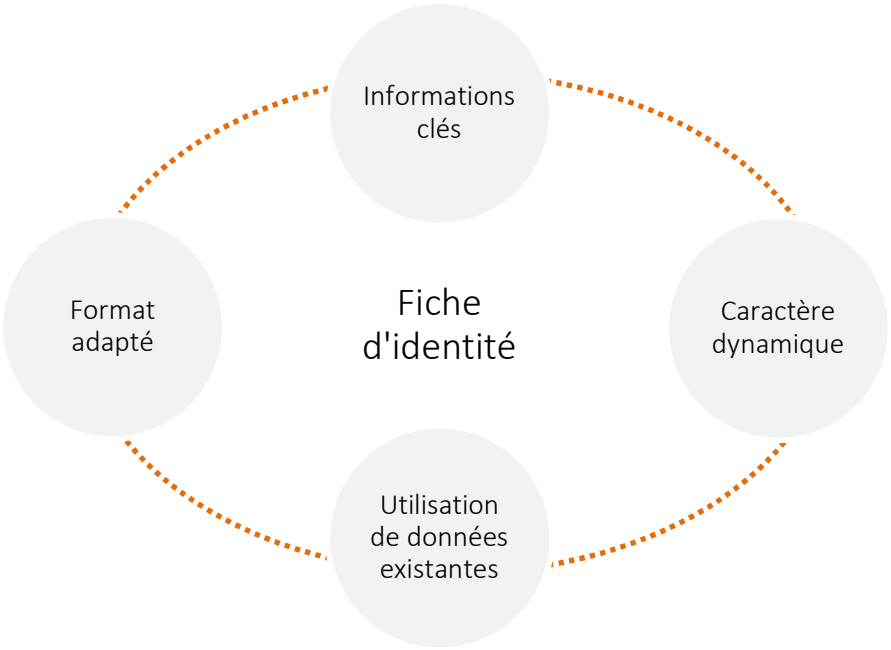


Figure 6: Schéma résumant les caractéristiques principales de la fiche d'identité de piézomètre

La sélection de l'information s'est principalement basée sur des réflexions pratiques guidées par l'avis des collègues qui travaillent régulièrement avec les données de piézomètres et doivent effectuer des mesures sur le terrain. Lors de plusieurs consultations communes, des propositions ont été faites, discutées, modifiées et ensuite validées. Cette mise au point permettait de définir le contenu souhaité pour la fiche d'identité et d'adapter la réalisation en fonction des commentaires obtenus.

Le caractère dynamique de ces données ainsi que la nécessité de disposer de données à jour ont également été mis en avant lors de ces consultations communes. Dans ce contexte, une mise à jour régulière des fiches d'identité semblait être la solution idéale, permettant d’assurer que les informations présentes sur les fiches étaient actuelles et correspondaient à l'information la plus récente disponible au GESDEC.

Les données pour la fiche proviennent principalement directement du GESDEC. Le système d’informations SOLSTISS constitue la source primaire de données et est complété par les informations additionnelles disponibles au sein du service. Le but est de se servir de ressources déjà existantes et de les mettre en valeur plutôt que de créer de la nouvelle information. Ceci permet également de respecter l'organisation des données au sein du service. Les données sont soit directement utilisées ou bien prétraitées avant l'utilisation afin de correspondre aux besoins formulés pour la fiche d'identité.

Le format de la fiche devrait permettre de répondre aux différents besoins du métier. Elle devrait donc à la fois être disponible en format numérique et sur papier afin de répondre au besoin de pouvoir être imprimée et de bien s'afficher sur l'ordinateur ainsi que sur des téléphones portables. En effet, cela permettrait de faciliter l'utilisation de cette fiche tant dans les bureaux que sur le terrain. Cela aurait l’avantage de de s'adapter aux préférences de chaque utilisateur·trice. Ainsi, le format PDF a été sélectionné, étant donné qu’il correspondait à ces exigences et permettait la valorisation de ces données dans un format pratique à l'emploi, qui ne demande pas d'adaptation pour la visualisation sur différents supports.

CATÉGORIES DE DONNÉES

Le but premier de ce projet de stage, était de mieux valoriser les données existantes sur les piézomètres. Dans une première partie, toutes les géodonnées utilisées pendant le processus de travail sont introduites. Ceci inclut également les données qui ont été utilisées dans le prétraitement des données uniquement. Dans une deuxième partie, toutes les catégories de données ainsi que les différents types de formats utilisés pour la création des fiches sont présentés.

GÉODONNÉES

Au cours du travail, les géodonnées ont été directement utilisées pour renseigner la fiche, mais ont également servi comme donnée « input » pour les géotraitements requis pour la préparation des données. Le tableau 1 résume les couches utilisées dans le cadre de ce travail et fournit des informations sur leur contenu et leur provenance.

Tableau 1: Détails sur les géodonnées utilisées pour le projet de stage : le nom de la couche, le contenu, le type de données, la source et le type d'accès (Source : crée par l'auteure, inspiré par République et Canton de Genève(2020b)).

Thématique	Nom de la couche	Contenu	Type de données	Source	Accès
Informations géologiques	GEOL_SS_SONDAGE	Information techniques sur les différents sondages dans le canton de Genève et dans l'agglomération	Classe d'entité vectorielle (point)	SOLSTISS	Restreint*
	GEOL_SS_ADMIN	Informations pour la gestion de données en interne (Identifiants etc.)	Table attributaire relationnelle	SOLSTISS	Restreint*
	GEOL_SS_EQUIPEMENT	Informations techniques sur les équipements des sondages	Table attributaire relationnelle	SOLSTISS	Restreint*
	GEOL_SS_SONDAGE_MESURE	Mesure effectuées à travers les équipements de sondages	Table attributaire relationnelle	SOLSTISS	Restreint*
	GEOL_SS_RESSOURCE	Information sur les différents types de ressources identifiées (hydrogéologiques).	Table attributaire relationnelle	SOLSTISS	Restreint*
Localisation (Adresse)	A.CAD_ADRESSE	Adresses géoreferencées pour les entrées de bâtiments dans le canton de Genève	Classe d'entité vectorielle (point)	SITG (DIT)	libre
	A.GMO_ROUTIER	Couche illustrant le graphe de la mobilité routière, illustrant les routes et les jonctions dans le canton de Genève	Classe d'entité vectorielle (polylignes)	SITG (DIT)	libre
	A.AGGLO_COMMUNES	Comunes suisses et françaises dans le canton de Genèves et dans l'agglomération vaudoise et française (Vaud, Ain, Jura, Haute-Savoie)	Classe d'entité vectorielle (polygones)	SITG (DIT)	libre
	A.GEO_ADRESSE_AGGLO	Adresses géoréférencées dans le canton de Genève et dans l'agglomération vaudoise et française (Vaud, Ain, Jura, Haute-Savoie).	Classe d'entité vectorielle (point)	SITG (DIT)	libre
Cartes	A.CAD_LIMITE_CANTON	Frontière administratives du canton de Genève (cantonales et nationales)	Classe d'entité vectorielle (polyligne)	SITG (DIT)	libre
	A.GEO_LAC_LEMAN	Représentation de l'emprise du lac Léman, du Rhône et de l'Arve	Classe d'entité vectorielle	SITG (DIT)	libre
	World Topographic Map	Basemap de ArcGIS	Basemap	esri	libre

* Accessible aux collaborateurs-trices du GESDEC

Les géoinformations géologiques proviennent de SOLSTISS, notamment de GEOL_SS_SONDAGE, une classe d'entité vectorielle contenant des points de sondage géoréférencés, ainsi qu'une sélection de tables attributaires relationnelles. GEOL_SS_ADMIN contient les informations liées à l'organisation des données tel que les identifiants. GEOL_SS_EQUIPEMENT indique les équipements liés aux sondages ainsi que leurs caractéristiques techniques. L'information de cette table permet de distinguer si un sondage est équipé d'un piézomètre. GEOL_SS_SONDAGE_MESURE contient toutes les informations relatives aux mesures réalisées par les équipements, tel que la date et le résultat de la mesure. La table GEOL_SS_RESSOURCE contient des données sur les types de ressources hydrogéologiques identifiées.

Pour rajouter à la géolocalisation du sondage une adresse en plus des coordonnées GPS, un prétraitement basé sur des géodonnées a été effectué (cf. Prétraitement des données). A cette fin, les données d'adresse du canton de Genève et celle de l'agglomération ont été utilisées (A.CAD_ADRESSE, A.GEO_ADRESSE_AGGLO), ainsi que le graphe de la mobilité routière comprenant les routes du canton (A.GMO_ROUTIER) et les délimitations communales dans l'agglomération de Genève (A.AGGLO_COMMUNES).

Pour la cartographie, un certain nombre de couches a été utilisé pour représenter les données dans des cartes. Il s'agit notamment de la délimitation du canton de Genève (A.CAD_LIMITE_CANTON), la représentation du lac Léman, du Rhône et de l'Arve (A.GEO_LAC_LEMAN) ainsi qu'une carte de base (World Topographic Map) qui a été retravaillée pour convenir aux besoins de ce travail.

SOURCES ET TYPES DE DONNÉES DISPONIBLES

Cette partie présente dans les grandes lignes les données recensées de tout type de format, qui sont d'intérêt pour la fiche d'identité de piézomètre. Deux principales sources de stockage de données ont été utilisées : la base de données de SOLSTISS et le serveur du GESDEC. Une représentation schématique, permettant de résumer les principales sources et catégories de données est disponible dans la figure 7.

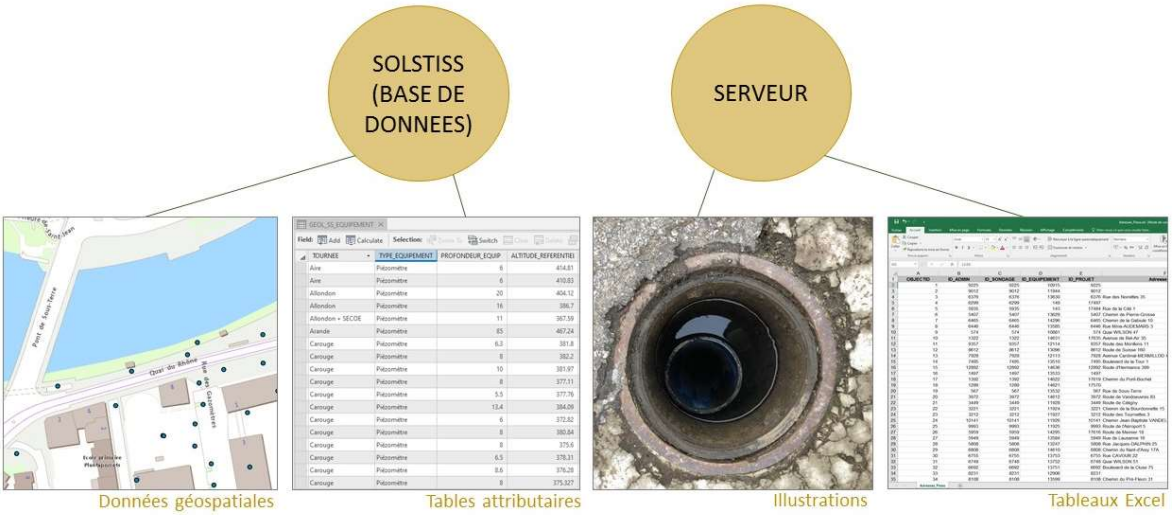


Figure 7: Représentation schématique des principales sources de données et catégories de données pertinentes pour la fiche d'identité de piézomètre

SOLSTISS fournit toutes les données géospatiales, tel que la géolocalisation ainsi que des tables attributaires relationnelles introduites précédemment (cf. tableau 1). Sur le serveur, un répertoire d'illustrations permet de rassembler toutes les images qui seront intégrées à la fiche. Ceci concerne des illustrations déjà existantes, telles que les photos de piézomètres disponibles au sein du service, ainsi que des schémas et des cartes de localisations générées lors du processus de création de la fiche. De plus, des tableaux élaborés pendant le prétraitement des données (cf. Prétraitement des données) et permettant de compléter les informations attributaires, ont été enregistrés sur le serveur. Pour des raisons de cohérence, il serait préférable que toutes les informations proviennent directement de SOLSTISS. Lorsque cela était possible, il a donc strictement été évité de créer une redondance en créant des nouvelles informations en-dehors de la base de données. Des informations externes ont uniquement été utilisées ou crées où ceci était absolument nécessaire car elles n'existaient pas (encore) dans SOLSTISS ou leur intégration était impossible.

Une liste exhaustive des données requises pour l'exécution du script est disponible dans la partie sur le script Python (cf. tableau 2).

LOGICIELS

Le traitement de données s'est principalement effectué sur ArcGIS Pro et à l'aide de Python sur ArcGIS Pro IDLE ainsi que partiellement sur Microsoft Excel.

Dans un premier temps, les différentes options de création d'un rapport en format PDF ont été explorées. ArcGIS propose l'option de créer un rapport à travers la fonctionnalité "Insert: New Report". Des valeurs statistiques sur les données peuvent ainsi être crées et insérées dans la fiche comme texte statique ou dynamique par exemple. Cependant les modalités de personnalisation restent très limitées et sont principalement destinées à la création manuelle du rapport. Des options plus avancées pour la création de PDF sont proposées pour la gestion de données émanant de sondages par exemple, dans le cadre de produits comme ArcGIS Survey123. Auparavant, l'option des CrystalReports était également proposée avec ArcGIS, mais elle n'est plus comprise à partir de ArcGIS 9.3 dans ArcGIS for Desktop (esri, s.d.).

Tenant compte du besoin d'automatisation de la création de la fiche (en raison du nombre de rapports à générer ainsi que de l'actualisation régulière), le choix a été porté sur la création d'un script Python, qui utilise l'extension ReportLab. ReportLab permet d'automatiser la création de documents PDF et offre également diverses options pour l'intégration de graphiques. L'utilisation de Python a permis d'automatiser simultanément le traitement des données et la création des fiches dans un même code, ce qui représente un avantage indéniable. ReportLab est une extension proposée dans le Python Package Manager dans ArcGIS Pro et la version basique est disponible gratuitement (cf. Détails sur l'installation disponible dans l'Annexe 3).

PRÉTRAITEMENT DES DONNÉES

Pour assurer que les données soient compatibles avec la suite des opérations, un certain nombre de prétraitements ont été nécessaires. Ces prétraitements consistaient en la création ou l'assemblage de nouvelles données requises pour la création de la fiche ainsi que leur formatage, notamment de tableaux de données et d'illustrations.

TABLEAUX : CRÉATION DE DONNÉES

Tel que mentionné précédemment, l'utilisation de données existantes a été privilégié. Cependant, certaines informations n'étaient pas disponibles dans SOLSTISS ni sur le serveur, raison pour laquelle deux nouveaux tableaux ont été générés. A terme, il serait désirable que ces informations soient intégrées aux tables attributaires de SOLSTISS, mais d'ici là ces tableaux seront utilisés pour la création des fiches.

Adresses

Pour la fiche d'identité du piézomètre, il semblait pertinent d'avoir en sus des coordonnées GPS, une adresse permettant de rapidement localiser un piézomètre. Dans SOLSTISS, il existe une adresse attribuée aux projets, mais cet attribut s'est avéré faiblement renseigné dans la base de données. De plus, les adresses disponibles n'étaient pas forcément très précises, étant donné que certains projets avaient une importante étendue géographique, parfois de plusieurs parcelles.

Pour obtenir une adresse plus viable pour chaque piézomètre, il a été décidé de réaliser une analyse spatiale à l'aide des outils ArcGIS «Near » et « Intersect ». Les géodonnées utilisées dans le cadre de ce prétraitement sont une couche représentant les piézomètres[1] ainsi que les données présentées dans la partie géodonnées sous l'onglet thématique « Localisation (Adresse) ». Cette analyse attribue les piézomètres à trois catégories en fonction de trois conditions successives. Ces conditions permettent de déterminer une adresse pour chaque piézomètre (détaillé et illustré dans la figure 8). Comme périmètre, il a été décidé de fixer 50 mètres comme limite, étant donné que ceci est assez précis pour visualiser l'emplacement et permet tout de même d'attribuer une adresse au plus grand nombre de piézomètres.

[1] Cette couche est générée en effectuant une sélection sur la couche GEOL_SS_SONDAGE en fonction du type d'équipement précisé dans GEOL_SS_EQUIPEMENT (= piézomètre)

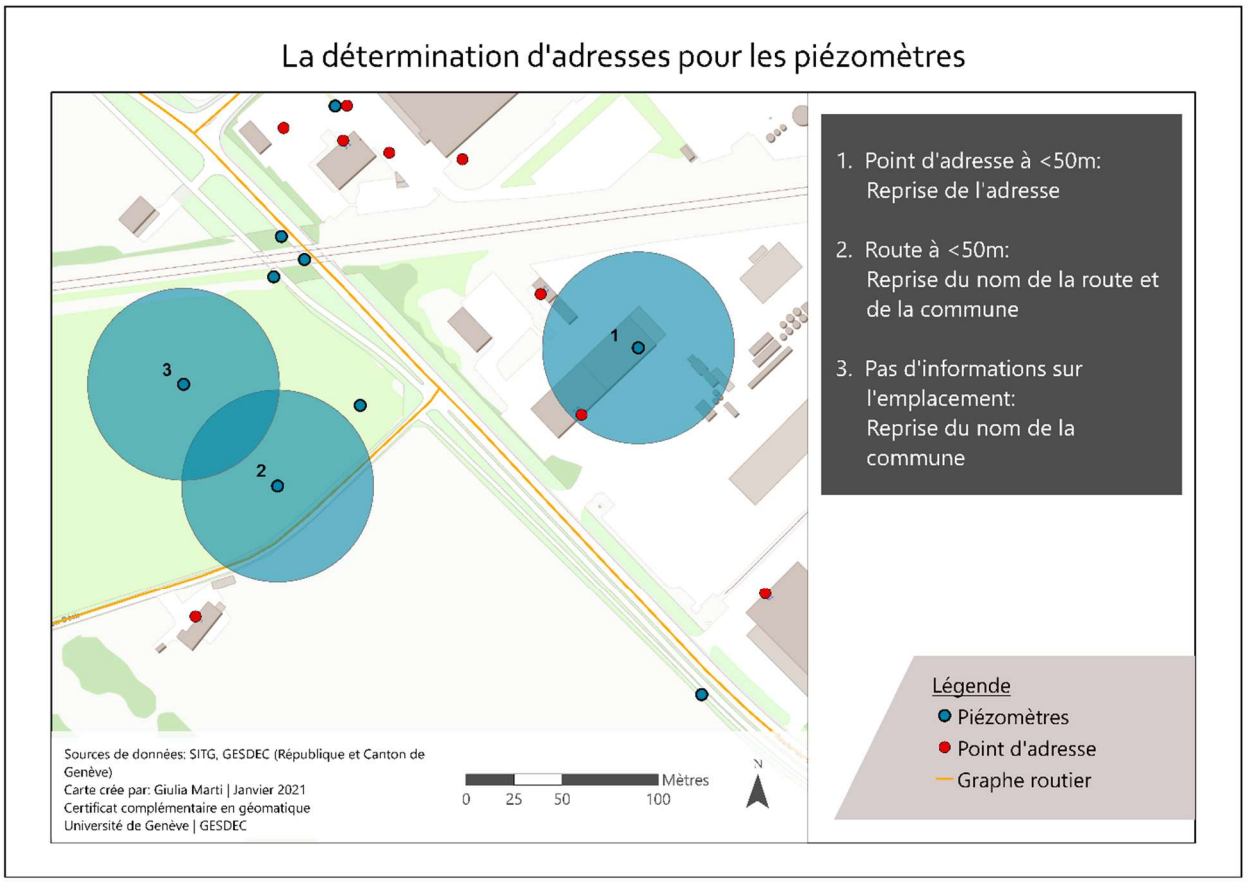


Figure 8: Critères pour l'attribution d'adresse à chacun des piézomètres. Si la condition 1 n'est pas remplie, la condition 2 s'applique. Idem pour les conditions 2 et 3.

Dans un premier temps, il est vérifié si un piézomètre fait partie de la catégorie 1. A l'aide de l'outil « Near », il est vérifié s'il existe un point d'adresse dans un périmètre de 50 mètres du piézomètre (A.CAD_ADRESSE). Si ceci est le cas, l'adresse de ce point d'adresse peut être reprise comme adresse du piézomètre. Une sélection est donc effectuée pour appliquer l'opération à tous les piézomètres de la catégorie 1, puis l'information est rajoutée à l'aide d'un « Join Field » depuis A.CAD_ADRESSE (Attributs : "ADRESSE", " COMMUNE", NO_POSTAL). Cette opération permet d'attribuer une adresse à plus de 70% des piézomètres (3644 piézomètres).

Dans un deuxième temps, il est nécessaire de déterminer la catégorie des piézomètres qui ne remplissaient pas la condition 1 : Si une route se trouve dans un périmètre de 50 mètres, les piézomètres sont attribués à la catégorie 2. Leur adresse est déterminée en se basant sur le graphe de la mobilité routière. Le nom de la rue la plus proche est reprise comme adresse. Un « Join Field » permet de transférer l'information depuis A.GMO_ROUTIER (Attribut : "NOMVOIE").

Finalement, uniquement le nom de la commune est retenu comme adresse pour les piézomètres restants appartenant à la catégorie 3. Une intersection entre la couche des

piézomètres et celle des communes (A.AGGLO_COMMUNES) est donc réalisée afin de distinguer dans quelle commune chacun des piézomètres se situe. Etant donné que les piézomètres de catégorie 2 n'ont pas d'information sur la commune mais uniquement un nom de route comme adresse, l'information de la commune est également reprise pour ces derniers. L'information sur la commune ainsi que le numéro de commune est conservé. Etant donné que la couche A.AGGLO_COMMUNES ne met pas à disposition le numéro postal, l'information est reprise depuis A.GEO_ADRESSE_AGGLO à l'aide du numéro de commune. Il fallait donc tenir compte du système de numérotation différent entre les communes suisses et françaises.

Pour harmoniser l'organisation des données, deux colonnes ont été créés : Une première avec l'adresse et une deuxième contenant le numéro postal et le nom de la commune. Ces attributs (y compris le numéro d'équipement servant comme identifiant) sont exportés en tant que tableau Excel. Le fichier est enregistré sur le serveur, permettant une utilisation facilitée de ces informations d'adresse.

Cependant, il faut garder en tête que la précision de l'adresse est de 50 mètres. Il est donc possible, que l'adresse indiquée pour le piézomètre ne soit pas forcément la « bonne » adresse. Ce cas de figure est par exemple illustré dans la figure 9 : Le point d'adresse du prochain bâtiment est plus proche (13.59m) et sera donc retenu, alors que l'adresse véridique serait différente (à une distance de 16.61m).

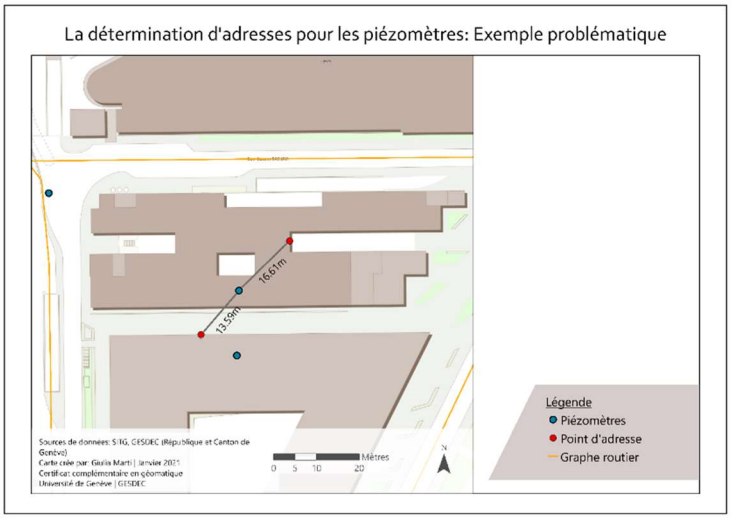


Figure 9: Cas de figure représentant une imprécision liée à la méthodologie choisie pour la détermination des adresses.

Cependant, malgré le degré de précision limité, l'indication d'une adresse est estimée comme bénéfique pour les utilisateurs·trices afin de visualiser la zone approximative de l'emplacement du piézomètre.

Sondes automatiques (Veille hydrologique)

Le GESDEC dispose de plusieurs piézomètres équipés d'une sonde permettant d'effectuer des mesures automatiques. Ces mesures des eaux souterraines peuvent être consultées en

temps réel sur le site internet de la veille hydrologique du Canton de Genève (https://vhg.ch). Pour les piézomètres en question, la fiche devrait donc inclure un renvoi vers ces mesures sur le site internet. Dans SOLSTISS, les piézomètres équipés d'une telle sonde automatique ne disposaient cependant pas encore d'attribut permettant de les distinguer. Ainsi, un fichier Excel a été créé et les numéros de piézomètres ont été recensés manuellement, afin de disposer d'une liste indiquant pour chacun des piézomètres la présence ou l'absence d'une sonde automatique. Cette information permet d'insérer dans les fiches d'identité un renvoi vers les mesures instantanées du site internet pour les piézomètres en question. Etant donné que le GESDEC prévoit d'équiper davantage de piézomètres avec une telle sonde, le fichier Excel devra être actualisé en fonction.

ILLUSTRATIONS

Afin de pouvoir intégrer des illustrations à la fiche, une réorganisation ainsi que la création de certaines représentations devaient être réalisées.

Photos

Dans un premier temps, toutes les photos disponibles au sein du service ont dû être répertoriées dans un seul dossier, qui a également été complété par des photos prises lors de sorties sur le terrain. La structure de l'organisation des photos devait être revue afin d'instaurer une systématique qui permet de retrouver les photos appartenant à un piézomètre en particulier.

Les photos ont donc été manuellement renommées, reprenant le numéro d'équipement ainsi que la perspective de la photo dans le nom du fichier. _1 était utilisé pour une prise de vue de loin et _2 pour une prise de vue macro du tube du piézomètre. Sur la gauche de la figure 10, se trouve par exemple la photo portant le nom « P351_1.jpg » et sur la droite « P351_2.jpg » pour le piézomètre avec le numéro d'équipement 351.

Figure 10: Exemple de photos et de leurs noms attribués pour le piézomètre avec le numéro d'équipement 351 : Gauche P351_1 et droite P351_2. (Source des photos: Giulia Marti 2021, prises pour le GESDEC)

De plus, la taille des photos était hétérogène, ce qui s'est avéré problématique lors de l'insertion dans la fiche. La taille des images a donc été uniformisées à l'aide d'un script Python (cf. script 1). Le script permet de réajuster la taille des photos et de les enregistrer dans le fichier répertoire.

Script 1: Script « Python ResizeImage.py » permettant d'uniformiser la taille des photos

```
#-*- coding: utf-8 -*-
# ---------------------------------------------------------------------------
## ResizeImage.py
# Description: Script pour l'adaptation de la taille des photos pour correspondre
aux exigences du script CreationFiche_EquipementSOLSTISSMesures.py.
# Il utilise comme input les images enregistrées dans le fichier Original puis les
transforme et les enregistre dans le fichier Resized.
# ---------------------------------------------------------------------------
### SCRIPT FOR ADAPTING THE SIZE OF THE PICTURES FROM THE PIEZOMETRES
print("-----------------------------------------------------------------------------")
print("SCRIPT FOR ADAPTING THE SIZE OF THE PICTURES FROM THE PIEZOMETRES")
print("-----------------------------------------------------------------------------")

### 1. IMPORT MODULES
print ("Import all required modules")
import arcpy
from PIL import Image
#Overwrite Output: permission to overwrite existing data
arcpy.env.overwriteOutput = True

# ---------------------------------------------------------------------------
### 2. DATA MANAGEMENT

# Input Data
inputPath =
"S:\\UO5196\\80_SYS_INFORMATION\\SI_GEOLOGIE\\05_DGSI\\01_Applications_Metiers\\SI_Geo
logie\\Stages\\Fiche_identite_piezo\\01_FichesPIEZO\\02_RepertoirePhotos\\Original"
outputPath="S:\\UO5196\\80_SYS_INFORMATION\\SI_GEOLOGIE\\05_DGSI\\01_Applications_Meti
ers\\SI_Geologie\\Stages\\Fiche_identite_piezo\\01_FichesPIEZO\\02_RepertoirePhotos\\R
esized"

# ---------------------------------------------------------------------------
### 3. TRANSFORMATION

# Define the workspace
arcpy.env.workspace = inputPath .
# List the files in the workspace
photoFiles= arcpy.ListFiles()

# Define the standard height the photos should have after the transformation
baseheight= 1500

# Loop through all the photos found in the input folder (inputPath)
for photo in photoFiles:
    print("Processing " + str(photo))
    # Define the exact path to the picture being processed
    photoPath = arcpy.env.workspace + "\\" + str(photo)
    # Opening the picture in order to modify the size
    img= Image.open(photoPath)
    # Determine the ratio of the desired height towards the original measures of the
    picture
    hpercent= (baseheight/float(img.size[1]))
    # This ratio allows to compute the new width of the picture, which allows to
    maintain the aspect ratio
    wsize= int((float(img.size[0])*float(hpercent)))
    # Resizing of the pricture according to the desired size of the picture
    img=img.resize((wsize, baseheight), Image.ANTIALIAS)
    # Determine the location in which the new picture should be saved
    output = outputPath + "\\" +str(photo)
    # Save the image
    img.save(output)

print("The size of all photos has been successfully adjusted").
```

Afin de pouvoir intégrer de nouvelles photos au processus de création de fiches, il est donc primordial de respecter l'organisation des noms de fichiers décrite auparavant et de faire tourner le script permettant d'ajuster leur taille (cf. Annexe 3).

Schéma

Afin de représenter les informations techniques des piézomètres à l'aide d'un schéma, deux versions de coupes simplifiées ont été dessinés (cf. figure 11). Le premier schéma représente un piézomètre qui ressort du terrain naturel et où le référentiel est situé au-dessus de ce dernier. Le deuxième schéma représente un piézomètre qui se situe complètement dans le sol et où le référentiel se trouve en-dessous du terrain naturel. Ces schémas ont été enregistrés en tant qu'images PNG. Ils peuvent être renseignés par les informations disponibles dans la base de données, tel que l'altitude du terrain, l'altitude du référentiel, la différence entre les deux altitudes et le diamètre du tube et du forage.

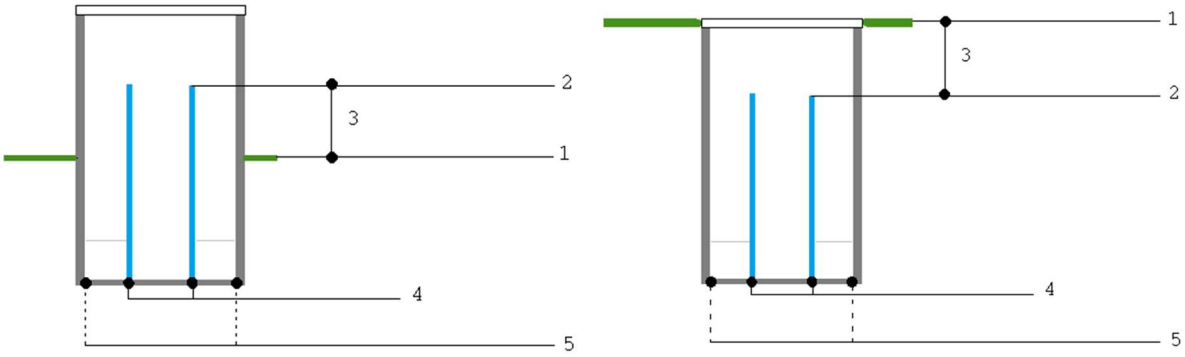


Figure 11: Schémas représentant deux types de piézomètres : Sur la gauche, où le référentiel se trouve au-dessus de l'altitude du terrain et sur la droite, où le référentiel se trouve en-dessous Les informations sont rajoutées lors de la création de la fiche : 1. Altitude du terrain, 2. Altitude du référentiel, 3. Différence entre les altitudes, 4. Diamètre du tube, 5. Diamètre du forage.

Carte: Adaptation de la basemap

En ce qui concerne la carte de fond (basemap) utilisée pour créer une carte de localisation, il s'est avéré que la lisibilité des annotations était faible (cf. figure 12). Tenant compte de l'importance de pouvoir lire le nom des rues à proximité, il était donc nécessaire d'agrandir les annotations.

La modification de caractéristiques des cartes de fond n'est pas possible dans ArcGIS Pro. Comme alternative, la Vector Style Editor App (https://developers.arcgis.com/vector-tile-style-editor/) proposé par esri a été la solution. Cette application permet de choisir une carte de fond

basique et d'ajuster une multitude de caractéristiques. Les annotations de la World Topographic Map ont donc été agrandies jusqu'à ce qu'elles soient lisibles dans la carte de localisation du piézomètre. Cette couche modifiée a pu être enregistrée sur ArcGIS Portal sur le compte utilisateur personnel et peut être intégré à des projets ArcGIS Pro depuis Portal. Cependant il n'est pas possible de la télécharger. Afin de l'utiliser, il est nécessaire d'obtenir les droits de la part des SITG (DIT) ou bien qu'elle soit partagée avec la personne par son/sa propriétaire.

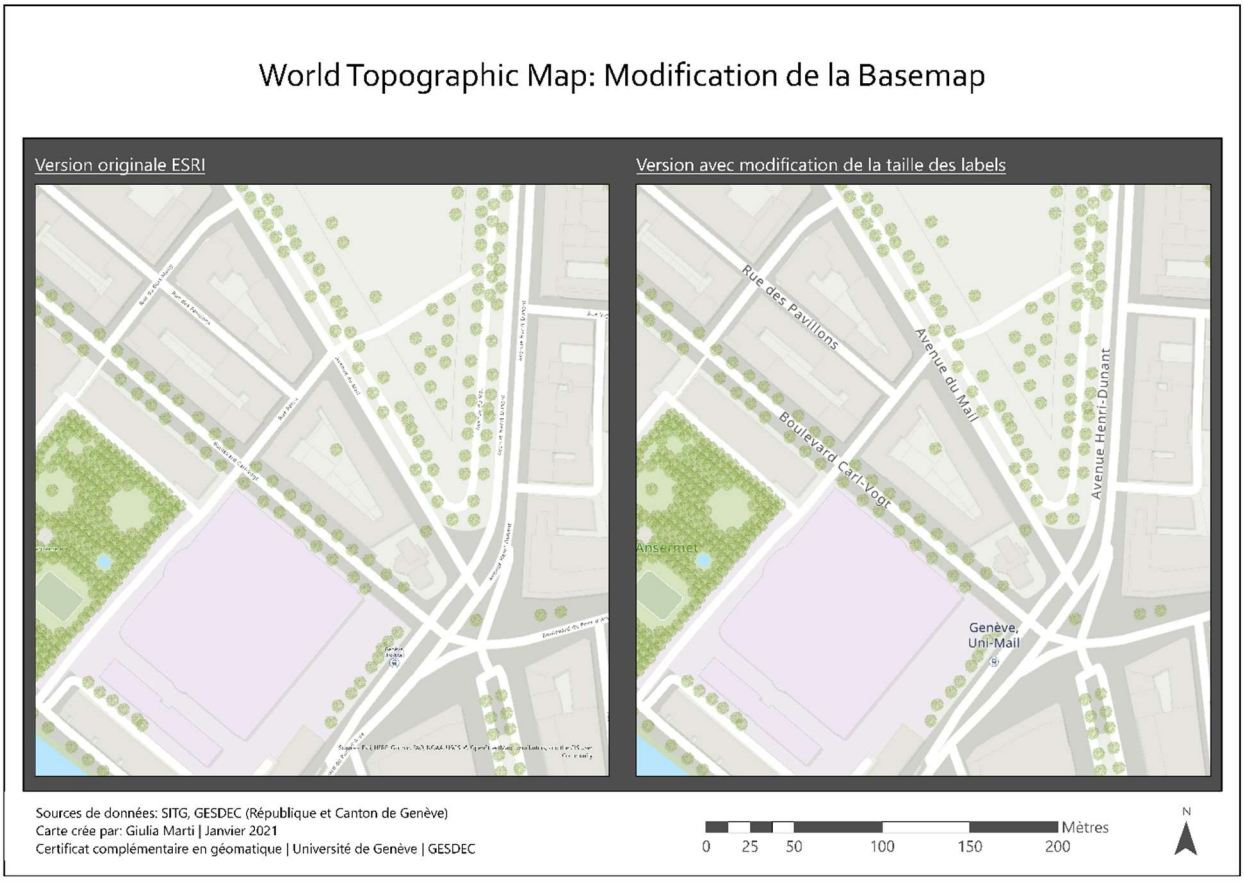


Figure 12: Comparaison entre la version originale de la World Topographic World Map sur la gauche et la version modifiée avec les annotations agrandies sur la droite. Cette modification a été réalisée à l'aide de la Vector Style Editor App.

SCRIPT PYTHON POUR LA CRÉATION DES FICHES D'IDENTITÉ

Le but est de construire un script Python qui permet de créer les fiches de piézomètre de manière automatique lors de son exécution. Cette partie méthodologique décrit les données requises en entrée (input) pour l'exécution du script et les différentes étapes du script Python en vue d'arriver au résultat final : les fiches d'identité des piézomètres en format PDF.

Pour des raisons de lisibilité, uniquement certains aperçus de code ont été intégrés au texte. Le script dans sa totalité peut être consulté dans les annexes.

DONNÉES INPUT POUR LE SCRIPT

Dans un premier temps, toutes les données input requises à l'exécution du script sont brièvement introduites, représentant un prérequis pour faire tourner le script Python. Le tableau 2 présente un résumé indiquant leurs noms précis, leur format et contenu ainsi que le type de données utilisés. Il s'agit de données du système d'information de SOLSTISS, ainsi que de fichiers Excel, dossiers d'illustrations ainsi que de projets ArcGIS Pro enregistrés sur le serveur.

Tableau 2: Résumé des données requises en input pour l'exécution du script Python permettant la création des fiches.

Source de stockage	Nom du fichier/de la couche	Format	Contenu	Type de données
SOLSTISS	Données géospatiales			
	GEOL_SS_SONDAGE	Classe d'entité vectorielle (point)	Information techniques sur les différents sondages dans le canton de Genève et dans l'agglomération	Points géoréférencés et attributs
	Tables attributaires			
	GEOL_SS_ADMIN	Table attributaire relationnelle	Informations pour la gestion de données en interne (Identifiants etc.)	Attributs
	GEOL_SS_SONDAGE_MESURE	Table attributaire relationnelle	Mesure effectuées à travers les équipements de sondages	Attributs
	GEOL_SS_EQUIPEMENT	Table attributaire relationnelle	Informations techniques sur les équipements des sondages	Attributs
	GEOL_SS_RESSOURCE	Table attributaire relationnelle	Information sur les différents types de ressources identifiées (hydrogéologiques).	Attributs
SERVEUR	ProjetTEST	Projet ArcGIS Pro (.aprx)	Projet dans lequel les données de SOLSTISS sont copiées avant chaque utilisation.	Points géoréférencés et tables attributaires (relationnelles)
	Tableaux			
	Adresses_Piezo	Fichier Excel (.xlsx)	Adresses pour les piézomètres du canton et de l'agglomération de Genève	Tableau de données
	SondeAutomatique	Fichier Excel (.xlsx)	Information si le piézomètre dispose du suivi automatique de la Veille hydrologique	Tableau de données
	Illustrations			
	Repertoire_Photos	Dossier	Dossier contenant les photos de piézomètres disponibles au sein du service	Photos au format JPEG
	Schema	Dossier	Dossier contenant deux types de schemas de piézomètres	Illustrations graphiques au format JPEG
	Piezometres_Map	Projet ArcGIS Pro (.aprx)	Basemap de ArcGIS pour laquelle les labels ont été modifiés (grand affichage des labels)	Vector tile basemap (disponible sur ArcGIS Portal *)
			Frontières cantonales Emprise du Lac Léman, Rhône et Arve	Classe d'entités vectorielles (polyligne et polygone)

* Accès limité aux personnes ayant obtenus les droits de consultation par le propriétaire de la couche ou les SITG

La première étape (1. IMPORT MODULES) consiste en l'importation des modules et de paramètres spécifiques ainsi qu'en la précision de certaines caractéristiques de l'environnement de travail (cf. script 2). Arcpy permet de faire le lien entre Python et les outils ArcGIS. Les modules os et sys permettent l'interaction avec le système opérateur. ReportLab permet la création des rapports PDF.

Script 2: Brève description du script "CreationFichePiezo_Serveur.py" et import des modules

```
#-*- coding: utf-8 -*-
# ---------------------------------------------------------------------------
## CreationFichePiezo_Serveur.py
# Description: Script pour la création d'une fiche d'identité pour les piézomètres.
Elle réunit une carte, un schema ainsi que toutes les caractéristiques techniques
connues sur le piézomètre.
# La principale source de donnée est la base de données SOLSTISS et certains autres
documents proviennent du serveur du GESDEC. Tous les liens sont axés sur le V: de
SOLSTISS ainsi que sur le serveur.
# Les fiches sont disponibles en format PDF sur le serveur du GESDEC après exécution
du script (Emplacement: cf. General Datapaths OUTPUT ci-dessous).
# ---------------------------------------------------------------------------
### SCRIPT FOR THE CREATION OF PIEZOMETRE IDENTITY FILES
print("----------------------------------------------------------------------------")
print("SCRIPT FOR THE CREATION OF PIEZOMETRE IDENTITY FILES")
print("----------------------------------------------------------------------------")

### 1. IMPORT MODULES
print ("Import all required modules")
# Import arcpy module
import arcpy
import os
import reportlab
from arcpy.sa import *
#Overwrite Output: permission d'écraser des données antécédentes
arcpy.env.overwriteOutput = True

# Import reportlab
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import A4, portrait
from reportlab.lib import colors
from reportlab import platypus
from reportlab.lib.styles import ParagraphStyle as PS
from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph

#Import PIL
from PIL import Image

#Import date
from datetime import date
```

La deuxième partie concerne la gestion des données (2. MANAGEMENT). Les liens précis vers les données input y sont définis (cf. script 3), ainsi que les liens vers des outputs intermédiaires (Working Layers) et vers les outputs finaux, à savoir les fiches d'identité.

Script 3: Définition des chemins d'accès vers les données input dans le script "CreationFichePiezo_Serveur.py"

```
# ---------------------------------------------------------------------------

### 2. DATA MANAGEMENT

#FOR THE COMPILATION TABLE
print("-------------------------------------------------------------------------------")
print ("Set the data input and output paths")
##Set the transferGDBAttributeProperties environment to True
arcpy.env.transferDomains = True

### General DataPaths INPUT
#Geodatabase SOLSTISS
dataPath=
"V:\Donnees_Applicatives\GEOMATIQUE\INTRANET\GEOTHERMIE2020_SOLSTISS\PROD\GEOTHERMIE2020_SOLSTISS.gdb\\"
#Links to ArcGIS projects
ProjetTEST =
"S:\\UO5196\\80_SYS_INFORMATION\\SI_GEOLOGIE\\05_DGSI\\01_Applications_Metiers\\SI_Geologie\\Stages\\Fiche_identite_piezo\\01_FichesPIEZO\\06_ProjetsArcGIS\\ProjetTEST\\"
ProjetTEST_GDB =ProjetTEST + "ProjetTEST.gdb"
Piezometres_Map =
"S:\\UO5196\\80_SYS_INFORMATION\\SI_GEOLOGIE\\05_DGSI\\01_Applications_Metiers\\SI_Geologie\\Stages\\Fiche_identite_piezo\\01_FichesPIEZO\\06_ProjetsArcGIS\\Piezometres_Map\\"

### General Datapaths OUTPUT
# Define the path to the ressources saved on the data server (Output)
dataserver =
"S:\\UO5196\\80_SYS_INFORMATION\\SI_GEOLOGIE\\05_DGSI\\01_Applications_Metiers\\SI_Geologie\\Stages\\Fiche_identite_piezo\\01_FichesPIEZO\\"
outputpath = dataserver + "05_OutputFiches\\"


### Input Data
#From SOLSTISS:
GEOL_SS_SONDAGE = dataPath + "GEOLOGIE_SOUS_SOL\\GEOL_SS_SONDAGE"
GEOL_SS_EQUIPEMENT = dataPath + "GEOL_SS_EQUIPEMENT"
GEOL_SS_ADMIN = dataPath + "GEOL_SS_ADMIN"
#GEOL_SS_ADRESSE = dataPath + "GEOL_SS_ADRESSE"
GEOL_SS_RESSOURCE = dataPath + "GEOL_SS_RESSOURCE"
#GEOL_SS_SONDAGE_MESURE: Problématique tant que la colonne "ID_EQUIPEMENT n'y figure pas dans le V:\. Dès qu'elle aura été activé, c'est la ligne de code ci-dessous qui devra être activé et celle du C:\ effacée.)
#GEOL_SS_SONDAGE_MESURE = dataPath + "GEOL_SS_SONDAGE_MESURE"
GEOL_SS_SONDAGE_MESURE =
"C:\\Users\MARTIGI\\AppData\\Roaming\\Esri\\ArcGISPro\\Favorites\\SOLSTISS.sde\\GEO2020.GEOL_SS_SONDAGE_MESURE"
#Manually created:
SONDE_AUTO = ProjetTEST + "SONDE_AUTOMATIQUE.xlsx\\Piezo_SondeAutomatique$"
ADRESSES_PIEZO = ProjetTEST + "Adresses_Piezo.xls\\Adresses_Piezo$"

#From the server:
# Define path to the schemata ("coupe")
schema1 = dataserver + "03_Schema\\AltEndessousTerrainMOD.png"
schema2 = dataserver + "03_Schema\\AltAudessusTerrainMOD.png"
# Define the path to the pictures
photos = dataserver + "02_RepertoirePhotos\\Resized"
maps = dataserver + "01_RepertoireMaps\\"
logo = dataserver + "02_RepertoirePhotos\\Logo\\GeneveNoirTransp.png"

#From the internet/server:
# Define paths to the geological surveys
sondagesprive =
"S:\\UO5196\\80_SYS_INFORMATION\\SI_GEOLOGIE\\FICHES_SONDAGES\\PRIVE\\"
sondagespublic = "https://ge.ch//sitg//geodata//SITG//SONDAGES_GEOLOGIQUES//"

#WorkingLayers (Assignation of names and paths for intermediate outputs)
Sondages = ProjetTEST_GDB  + "\\" + "Sondages"
Piezo = ProjetTEST_GDB  + "\\" + "Piezo" # Path to the compiled table in the GEODATABASE (Assembled from attribute tables, input data for the PDF File)
outputname = ProjetTEST + "Piezo_Copy.xlsx"
Sondages_Lyr = ProjetTEST_GDB + "\\" + "Sondages_Lyr"
Mesures_Copy = ProjetTEST_GDB + "\\" + "Mesures_Copy"
MESURE_Stat = ProjetTEST_GDB + "\\" + "MESURE_Stat"
Last_Measure_Date = ProjetTEST_GDB + "\\" + "Last_Measure_Date"
Date_Meas = ProjetTEST_GDB + "\\" + "Date_Meas"
GEOL_SS_EQUIPEMENT_table = ProjetTEST + "\\" +"GEOL_SS_EQUIPEMENT_table.xlsx"
GEOL_SS_EQUIPEMENT_tableF = ProjetTEST + "\\" + "GEOL_SS_EQUIPEMENT_table.xlsx" +
"\\" + "GEOL_SS_EQUIPEMENT_table$"
```

Précisions : La table attributaire GEOL_SS_SONDAGE_MESURE dans SOLSTISS migré sur V : ne dispose pas encore de l'attribut ID_EQUIPEMENT. Pour cette raison-là, la table de l'ancienne version de SOLSTISS est utilisée dans ce script. Ceci devrait être modifié prochainement, ce qui permettrait d'activer la ligne qui est grisée dans la section #InputData. De plus, GEOL_SS_ADRESSE pourra éventuellement être actualisé avec les adresses dans le fichier Excel, mais est actuellement désactivé dans le script.

TABLE SYNTHÉTIQUE

Tenant compte de la disparité des informations, il semblait pertinent de regrouper toutes les données attributaires concernant les piézomètres. Comme première étape, il est nécessaire de créer une nouvelle couche qui représente uniquement les piézomètres. Toutes les opérations sont effectuées dans le projet ArcGIS Pro « ProjetTEST.aprx » comme espace de travail, qui se trouve sur le serveur. La couche de départ, GEOL_SS_SONDAGE qui représente l'ensemble des sondages, est copiée-collée dans cet espace de travail. Pour parvenir à créer cette nouvelle couche de piézomètres, l'information sur le type d'équipement dans GEOL_SS_EQUIPEMENT est jointe à la couche GEOL_SS_SONDAGE. A travers une sélection basée sur le type d'équipement (=Piézomètre), un nouveau shapefile représentant uniquement les sondages équipés de piézomètres est créé (illustré dans la figure 4). Cette nouvelle couche est nommée « Piezo ».

Par la suite, les informations des autres tables attributaires relationnelles de SOLSTISS ainsi que les tableaux Excel du serveur sont rajoutés à la table attributaire de ce shapefile. La figure 13 représente les différentes tables attributaires et les tableau Excel utilisés, ainsi que les attributs de chaque table qui ont été repris. Pour relier les informations, deux attributs ont été utilisés : ID_ADMIN et ID_EQUIPEMENT. ID_ADMIN est l'attribut en commun entre GEOL_SS_SONDAGE, GEOL_SS_EQUIPEMENT, GEOL_SS_ADMIN et GEOL_SS_RESSOURCE. ID_EQUIPEMENT permet de relier GEOL_SS_EQUIPEMENT, GEOL_SS_SONDAGE_MESURE, et les tableaux Excel.

Pour toutes les tables attributaires à l'exception de GEOL_SS_SONDAGE_MESURE, les attributs peuvent directement être repris en indiquant l'attribut en commun sur lequel baser la jointure (« Join Field ») Le code est représenté dans le script 4. Pour certaines tables (p.ex. GEOL_SS_EQUIPEMENT, GEOL_SS_RESSOURCE), une première conversion en format Excel était cependant requise afin d'éviter qu'un problème avec les domaines surgisse (« domains » en anglais). Ceci est dû au fait que la base de données SOLSTISS utilise de nombreuses listes déroulantes à choix pour renseigner l'information standardisée et que la reprise de l'information non-codée n'était pas systématiquement possible. Ceci causait des problèmes en cas de jointure, car au lieu de l'information comprise dans le domaine, uniquement le chiffre codé s'affichait (p.ex. pour Etat de l'ouvrage : 2 au lieu de « Actif »). Il a donc été décidé d'exporter les tableaux intermédiaires en format Excel (« tableToExcel ») afin de contourner cette problématique. Les outputs intermédiaires sont effacés à la fin du processus. Il reste à préciser que la problématique des domaines persistait malgré l'option de l'environnement «arpy.env.transferDomains = True » activée dans le data management.

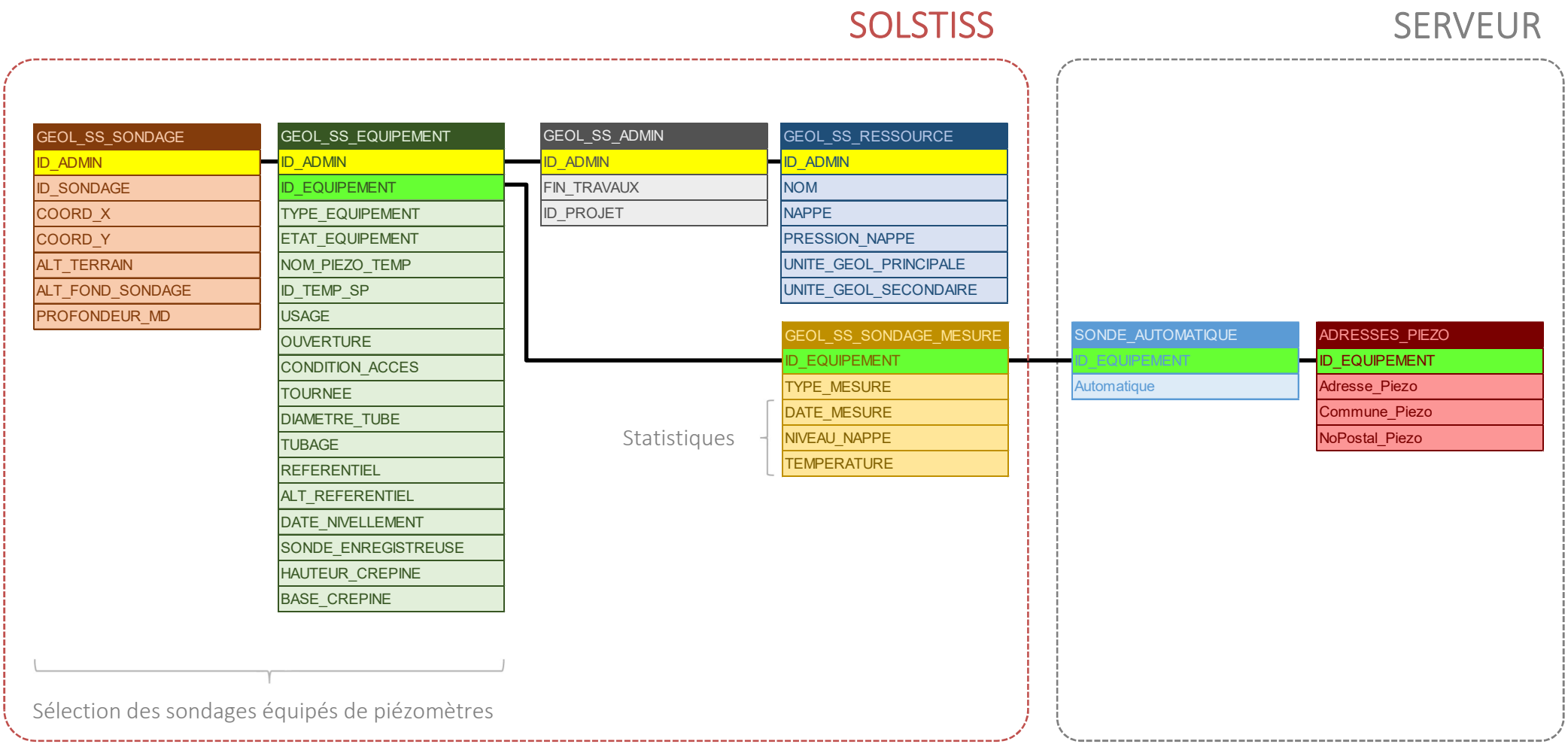


Figure 13: Représentation graphique des tables et attributs utilisés pour créer la table synthétique piézomètres ("Piezo"). Dans un premier temps les piézomètres parmi les sondages sont sélectionnés puis les informations des différentes tables sont regroupées basés sur les attributs en commun (ID_ADMIN ou ID_EQUIPEMENT).

Script 4: Création d'un shapefile et d'une table attributaire synthétique pour l'ensemble des piézomètres de la base de données dans le script "CreationFichePiezo_Serveur.py". Compilation de toutes les tables attributaires relationnelles, à l'exception de GEOL_SS_SONDAGE_MESURE, et tableaux Excel.

```
# ---------------------------------------------------------------------------
### 3. DATA PREPARATION: CREATION OF THE WORKING INPUT

##INFORMATION FROM THE DATASERVERS AND OTHERS: COMPILATION
print("-------------------------------------------------------------------------------")
print("Compilation of the relevant data for the input table")
#Define working environment
arcpy.env.workspace = ProjetTEST

#Recompilation of an Attribute Table for Piezometres

#Copy Piezo Shapefile to Working GDB
print ("- Copy Shapefile to Working GDB")
arcpy.management.CopyFeatures(GEOL_SS_SONDAGE, Sondages, '', None, None, None)

#Delete Irrelevant Fields for our purpose in the table serving as basic entity
print ("- Delete chosen Attributes")
arcpy.management.DeleteField(Sondages,
"OBJECTIF;METHODE;TECHNIQUE;ALT_FOND_SONDAGE;PROFONDEUR_TVD;UNITE_GEOLOGIQUE_ATTEINTE;
ORIENTATION;DATE_SAISIE;AUTEUR_SAISIE;DATE_MAJ;AUTEUR_MAJ;REMARQUE;DELTA_MODELE_TERRAI
N;VALEUR_MODELE_TERRAIN;GEOLOGUE_RESPONSABLE")

#Rajout de l'attribut qui permet de faire la sélection dans sondage:
print ("- Add Information from GEOL_SS_EQUIPEMENT")
arcpy.conversion.TableToExcel(GEOL_SS_EQUIPEMENT, GEOL_SS_EQUIPEMENT_table, "NAME",
"DESCRIPTION")
arcpy.management.JoinField(Sondages, "ID_ADMIN", GEOL_SS_EQUIPEMENT_tableF,
"ID_ADMIN", ["TYPE_EQUIPEMENT", "ID_TEMP_SP", "TOURNEE", "DATE_NIVELLEMENT",
"DISTANCE_A_PIEDS", "CONDITION_ACCES", "OUVERTURE", "USAGE",
"PROFONDEUR_EQUIPEMENT", "ALT_REFERENTIEL",

   "TUBAGE", "DIAMETRE_TUBE", "HAUTEUR_CREPINE", "BASE_CREPINE",
"SONDE_ENREGISTREUSE", "REFERENTIEL", "ID_EQUIPEMENT",
"ETAT_EQUIPEMENT","NOM_PIEZO_TEMP"])
arcpy.management.MakeFeatureLayer(in_features=Sondages, out_layer=Sondages_Lyr)

#Créer une sélection pour uniquement garder les piézomètres
arcpy.management.SelectLayerByAttribute(Sondages_Lyr, "NEW_SELECTION",
"TYPE_EQUIPEMENT = 'Piézomètre'", None)
arcpy.management.CopyFeatures(Sondages_Lyr, Piezo)

#Add Information from other Tables keeping the domains:
print ("- Add Information from GEOL_SS_ADMIN")
arcpy.management.JoinField(Piezo, "ID_ADMIN" , GEOL_SS_ADMIN, "ID_ADMIN",
["ID_PROJET", "FIN_TRAVAUX"])

#GEOL_SS_RESSOURCE
print ("- Add Information from GEOL_SS_RESSOURCE")
#Add Information for Fields where only number has been added (due to the domain,
circumvent by Table to Table):
arcpy.conversion.TableToTable(GEOL_SS_RESSOURCE, ProjetTEST_GDB, "CopyRessource")
CopyRessource = ProjetTEST_GDB + "\\" + "CopyRessource"
arcpy.management.JoinField(Piezo, "ID_ADMIN", CopyRessource, "ID_ADMIN", ["NOM",
"NAPPE", "PRESSION_NAPPE", "UNITE_GEOL_PRINCIPALE", "UNITE_GEOL_SECONDAIRE"])


#ADRESSES_PIEZO
print ("- Add Adresses from ADRESSES_PIEZO")
arcpy.management.JoinField(Piezo, "ID_EQUIPEMENT", ADRESSES_PIEZO, "ID_EQUIPEMENT",
["Adresse_Piezo", "NoPostal_Piezo", "Commune_Piezo"])

# ATTENTION: sera à terme evtl. intégré aux adresses de projet ci-après
(GEOL_SS_ADRESSE) et il faudra donc activer la ligne ci-dessous et retirer celle
en-haut (#ADRESSES_PIEZO)
#GEOL_SS_ADRESSE
#print ("- Add Information from GEOL_SS_ADRESSE")
#arcpy.management.JoinField(Piezo, "ID_PROJET" , GEOL_SS_ADRESSE, "ID_PROJET",
["ADRESSE_RUE", "ADRESSE_NUM_RUE", "ADRESSE_NPA", "ADRESSE_VILLE"])

#SONDE_AUTO
print ("- Add Information from SONDE_AUTO")
arcpy.management.JoinField(Piezo, "ID_EQUIPEMENT", SONDE_AUTO, "ID_EQUIPEMENT",
["AUTOMATIQUE", "LIEN_VH"])

#Delete intermediate outputs
print ("- Delete intermediate output")
arcpy.management.Delete(CopyRessource)
arcpy.management.Delete(GEOL_SS_EQUIPEMENT_table)
print ("Table has been compiled successfully")
```

La table de mesures GEOL_SS_SONDAGE_MESURE représente une exception, car les données ne pouvaient pas être directement reprises. Ceci est dû au fait que cette table regroupe toutes les mesures effectuées pour tous les piézomètres confondus. Afin de distinguer les informations qui nous intéressent, notamment le minimum, le maximum et la moyenne du niveau de la nappe ainsi que la dernière mesure et la moyenne de température mesurée pour chaque piézomètre, il était nécessaire d'effectuer un traitement de données plus avancé (cf. script 5). Cela signifiait notamment de réaliser des statistiques sur toutes les mesures de la table (« Summary Statistics »). Dans une première analyse, le niveau de nappe moyen, minimum et maximum ainsi que la moyenne de température et le nombre de mesures de niveaux d'eau et de température étaient déterminés en fonction de chaque piézomètre. Ensuite, afin de distinguer la dernière date de mesure, la fonction « MAX» de statistique est appliquée, en fonction du type de mesure (niveau d'eau ou ou température) et pour chaque piézomètre. Pour distinguer la dernière date de mesure pour le prélèvement du niveau d'eau, une sélection est appliquée à cette table et l'information est ensuite joint à la table de statistique initiale. Une difficulté rencontrée était notamment, que la fonction « LAST » de la statistique par ArcGIS ne donnait pas le résultat escompté pour les dates, bien que cela aurait dû rechercher la dernière mesure. Pour cette raison, la fonction « MAX » a été appliquée à la date, permettant de distinguer la dernière mesure.

Etant donné que nous souhaitions également connaître la date de la mesure minimale et maximale de la nappe (MIN_date et MAX_date) ainsi que le niveau d'eau de la dernière mesure (DER_NIVEAU_NAPPE), il était par la suite nécessaire de faire une boucle à travers l'ensemble des piézomètres, permettant de retrouver la date correspondante au minimum et maximum de nappe mesuré ainsi que le niveau d'eau correspondant à la dernière date de mesure pour chacun des piézomètres. Ce processus est effectué à travers la fonction arcpy.da.SearchCursor qui permet de parcourir un tableau de données en lecture et de retrouver les données correspondant à certains critères ainsi que arcpy.da.UpdateCursor qui permet de modifier le contenu des cellules dans le tableau. Ce processus est cependant très laborieux et a pris beaucoup de temps, dû à l'important nombre de mesures enregistrées.

Les attributs ainsi calculés, notamment le niveau moyen de la nappe, la mesure minimum et maximum ainsi que les dates respectives, le dernier niveau de nappe mesuré et la date respective, la moyenne de température et le nombre de mesures effectuées sont rajoutés à la couche « Piezo » (cf. Script 5).

Pour faciliter et accélérer la reprise des informations, la table attributaire de « Piezo » a été exportée comme tableau Excel « Piezo_Copy.xlsx » sur le serveur et a été utilisée par la suite comme unique input pour les données attributaires.

Script 5: Détermination des mesures d'intérêt pour chaque piézomètre (le minimum, maximum et la dernière mesure du niveau d'eau et les dates respectives, la moyenne de niveau d'eau, le nombre de mesures de niveau d'eau et de température ainsi que la moyenne de température) dans le script "CreationFichePiezo_Serveur.py". Ces données sont ensuite rajoutées à la table attributaire « Piezo », qui est finalement transformée en tableau Excel.

```
##DETERMINATION OF THE LASTEST MEASURES AND RECORDS
print("-------------------------------------------------------------------------------")
print("Determination of the latest measures and records in the database")
#Define working environment
arcpy.env.overwriteOutput = True
arcpy.env.workspace = ProjetTEST_GDB

#Creation of the Summary
arcpy.analysis.Statistics(GEOL_SS_SONDAGE_MESURE, MESURE_Stat, "NIVEAU_NAPPE
MEAN;NIVEAU_NAPPE MIN;NIVEAU_NAPPE MAX;TEMPERATURE MEAN; NIVEAU_MESURE
COUNT;TEMPERATURE COUNT", "ID_EQUIPEMENT")
arcpy.analysis.Statistics(GEOL_SS_SONDAGE_MESURE, Last_Measure_Date, "DATE_MESURE
MAX", "ID_EQUIPEMENT;TYPE_MESURE")
arcpy.analysis.TableSelect(Last_Measure_Date, Date_Meas, "TYPE_MESURE = 2")
arcpy.management.JoinField(MESURE_Stat, "ID_EQUIPEMENT", Date_Meas, "ID_EQUIPEMENT",
"MAX_DATE_MESURE")

#Creation of new columns to receive the data
arcpy.management.AddField(MESURE_Stat, "MIN_Date", "DATE")
arcpy.management.AddField(MESURE_Stat, "MAX_Date", "DATE")
arcpy.management.AddField(MESURE_Stat, "DER_NIVEAU_NAPPE", "FLOAT")


print("Data selection and definition")
fieldsStat = ["ID_EQUIPEMENT", "MIN_NIVEAU_NAPPE", "MAX_NIVEAU_NAPPE",
"MAX_DATE_MESURE", "MIN_Date", "MAX_Date", "DER_NIVEAU_NAPPE"]
fieldsMes =["ID_EQUIPEMENT", "DATE_MESURE", "NIVEAU_NAPPE", "TYPE_MESURE"]
def unique_values(table, field):
    with arcpy.da.SearchCursor(MESURE_Stat, "ID_EQUIPEMENT") as cursor:
        return sorted({row[0] for row in cursor})
piezos_list = unique_values(MESURE_Stat, "ID_EQUIPEMENT")

for i in piezos_list:
    expression = '"ID_EQUIPEMENT" = ' + str(i)
    # Grasp the data for each piezometre
    #First cursor
    with arcpy.da.SearchCursor(MESURE_Stat, fieldsStat, expression) as cursor:
        for row in cursor:
            ID_EQUIPEMENT = str(row[0])
            MIN_NAPPE = row[1]
            MAX_NAPPE = row[2]
            DERMESURE_Date = row[3] #Dernière mesure nappe

        #Search in data corresponding value
        if MIN_NAPPE is not None:
            MIN_NAPPE = str("{:.2f}".format(float(MIN_NAPPE)))
            expression1 = '"ID_EQUIPEMENT" = '+ str(i) + " AND " + '"NIVEAU_NAPPE" =
            ' + MIN_NAPPE
            list1=[] # Create a list with all the dates when the minimum date is
            being met
            with arcpy.da.SearchCursor(GEOL_SS_SONDAGE_MESURE, fieldsMes,
            expression1) as cursor:
                for row in cursor:
                    MIN_Date = row[1]
                    list1.append(MIN_Date)
            if list1: # should be not when the list is not emtpy
                latest_min= max(list1)
            else:
                latest_min = None

        if MAX_NAPPE is not None:
            MAX_NAPPE = str("{:.2f}".format(float(MAX_NAPPE)))
            expression2 = '"ID_EQUIPEMENT" = '+ str(i) + " AND " + '"NIVEAU_NAPPE" =
            ' + MAX_NAPPE
                        str("{:.2f}".format(float(DER_NIVEAU_NAPPE)))
        if DERMESURE_Date is None:
            DER_NIVEAU_NAPPE = None


        with arcpy.da.UpdateCursor(MESURE_Stat, fieldsStat, expression) as cursor:
            for row in cursor:
                row[4]=latest_min
                row[5]=latest_max
                row[6]=DER_NIVEAU_NAPPE
                cursor.updateRow(row)

#Join information from measures to the Piezo table
print ("- Add Information from MESURE_Stat")
arcpy.management.JoinField(Piezo, "ID_EQUIPEMENT", MESURE_Stat, "ID_EQUIPEMENT",
["MEAN_NIVEAU_NAPPE", "MIN_NIVEAU_NAPPE", "MIN_Date", "MAX_NIVEAU_NAPPE",
"MAX_Date", "DER_NIVEAU_NAPPE", "MAX_DATE_MESURE", "MEAN_TEMPERATURE",
"COUNT_NIVEAU_MESURE", "COUNT_TEMPERATURE"])


# TRANSFORMATION INTO AN EXCEL TO FACILITATE LATER WORK
# COPY THE GEODATA INFORMATION INTO EXCEL
# (otherwise problems with data linked to domains, providing the coded number
instead of the descriptions)
print("Copy of the table into Excel format")
arcpy.conversion.TableToExcel(Piezo, outputname, "NAME", "DESCRIPTION")
```

CARTE DE LOCALISATION

Afin d'illustrer l'emplacement d'un piézomètre, une carte de localisation devrait également figurer sur la fiche d'identité de piézomètre. Le défi était donc de créer un code Python permettant d'automatiser la génération d'une carte de localisation pour chacun des piézomètres. Ceci est réalisé à l'aide du module arcpy.mp de ArcGIS Pro. Le code est structuré en deux parties. Une partie générale qui est effectuée une fois, permettant de créer la carte de base et une partie spécifique, consistant en une fonction, qui est exécutée individuellement pour chaque piézomètre pendant la création des fiches.

Préparation de la carte générale

Dans la partie plus générale du script, effectuée uniquement une fois, la carte générale (« Basic Map ») est créée (cf. script 6). Cette carte est destinée à afficher la couche des piézomètres générée pendant la création de la table synthétique. Cette couche est donc toujours à jour, ce qui assure que les nouveaux piézomètres soient inclus dans la représentation lors d'une nouvelle exécution du script. Etant donné que ce module travaille uniquement avec des « layer files », le shapefile Piezo est transformé en feature layer (« Make Feature Layer ») et enregistré en tant que tel (Piezo_Layer = «Piezo.lyrx »).

L'environnement de travail pour la création des cartes est un projet ArcGIS Pro déjà existant : « Piezometres_Map.aprx ». Il comprend deux différents onglets de cartes (« Maps ») : une carte permettant de représenter l'emplacement du piézomètre à une échelle rapprochée (« Map ») et une carte d'aperçu représentant le canton (« Geneve », overview map). La carte « Map » contient comme carte de fond la World Topographic Map avec les annotations modifiées et la carte « Geneve » avec une couche vectorielle représentant le canton de Genève et une autre avec le Lac Léman, le Rhône et l'Arve. En plus, le projet contient une mise en page (« Layout2 ») déjà préparée, y compris l'arrangement des « data frames », la légende, la flèche indiquant le nord et l'échelle.

Dans une prochaine étape, le layer file Piezo_Layer est rajouté à la carte « Map » et est renommé pour la visualisation (« Piézomètres »). Le module permet également de modifier la symbologie à travers le script, notamment l'allure, la taille et la couleur du symbole. Les piézomètres de la carte générale sont configurés afin d'être représentés par des points bleus. Le projet actualisé est enregistré afin d'être prêt pour la suite des opérations.

Script 6: Code pour la préparation des données cartographiques des piézomètres pour la carte générale dans le script "CreationFichePiezo_Serveur.py"

```
## PREPARATION OF THE DATA FOR THE MAP
# Process: Make Feature Layer (Make Feature Layer)
print("------------------------------------------------------------------------------")
print ("Data Preparation for the map")
Output_Layer = "Piezo"
arcpy.MakeFeatureLayer_management(in_features=Piezo, out_layer=Output_Layer,
where_clause="", workspace="", field_info="OBJECTID OBJECTID VISIBLE NONE;SHAPE
SHAPE VISIBLE NONE;ID_EQUIPEMENT ID_EQUIPEMENT VISIBLE NONE;ID_SONDAGE ID_SONDAGE
VISIBLE NONE;NOM_PIEZO_TEMP NOM_PIEZO_TEMP VISIBLE NONE;TOURNEE TOURNEE VISIBLE
NONE;NO_PRIVE NO_PRIVE VISIBLE NONE;ALT_TERRAIN ALT_TERRAIN VISIBLE
NONE;PROFONDEUR_MD PROFONDEUR_MD VISIBLE NONE;ID_ADMIN ID_ADMIN VISIBLE NONE;COORD_X
COORD_X VISIBLE NONE;COORD_Y COORD_Y VISIBLE NONE;UNITE_GEOLOGIQUE_ATTEINTE
UNITE_GEOLOGIQUE_ATTEINTE VISIBLE NONE;DISTANCE_A_PIEDS DISTANCE_A_PIEDS VISIBLE
NONE;CONDITION_ACCES CONDITION_ACCES VISIBLE NONE;OUVERTURE OUVERTURE VISIBLE
NONE;USAGE USAGE VISIBLE NONE;PROFONDEUR_EQUIPEMENT PROFONDEUR_EQUIPEMENT VISIBLE
NONE;ALT_REFERENTIEL ALT_REFERENTIEL VISIBLE NONE;TUBAGE TUBAGE VISIBLE
NONE;DIAMETRE_TUBE DIAMETRE_TUBE VISIBLE NONE;HAUTEUR_CREPINE HAUTEUR_CREPINE
VISIBLE NONE;BASE_CREPINE BASE_CREPINE VISIBLE NONE;SONDE_ENREGISTREUSE
SONDE_ENREGISTREUSE VISIBLE NONE;REMARQUE REMARQUE VISIBLE NONE;DATE_NIVELLEMENT
DATE_NIVELLEMENT VISIBLE NONE;REFERENTIEL REFERENTIEL VISIBLE NONE;ID_EQUIPEMENT
ID_EQUIPEMENT VISIBLE NONE;ETAT_EQUIPEMENT ETAT_EQUIPEMENT VISIBLE NONE;ID_TEMP_SP
ID_TEMP_SP VISIBLE NONE;ID_PROJET ID_PROJET VISIBLE NONE;ADRESSE_RUE ADRESSE_RUE
VISIBLE NONE;ADRESSE_NPA ADRESSE_NPA VISIBLE NONE;ADRESSE_VILLE ADRESSE_VILLE
VISIBLE NONE;ADRESSE_NUM_RUE ADRESSE_NUM_RUE VISIBLE NONE;TYPE_CONTACT TYPE_CONTACT
VISIBLE NONE;ID_CONTACT ID_CONTACT VISIBLE NONE;RAISON_SOCIALE RAISON_SOCIALE
VISIBLE NONE;EMAIL EMAIL VISIBLE NONE;TELEPHONE TELEPHONE VISIBLE NONE;ADRESSE_VILLE
ADRESSE_VILLE VISIBLE NONE;NAPPE NAPPE VISIBLE NONE;PRESSION_NAPPE PRESSION_NAPPE
VISIBLE NONE;UNITE_GEOL_PRINCIPALE UNITE_GEOL_PRINCIPALE VISIBLE
NONE;UNITE_GEOL_SECONDAIRE UNITE_GEOL_SECONDAIRE VISIBLE NONE;REMARQUE_1 REMARQUE_1
VISIBLE NONE;Automatique Automatique VISIBLE NONE;Lien_VH Lien_VH VISIBLE
NONE;Adresse_Piezo Adresse_Piezo VISIBLE NONE;NoPostal_Piezo NoPostal_Piezo VISIBLE
NONE;Commune_Piezo Commune_Piezo VISIBLE NONE")

# Process: Save To Layer File (Save To Layer File)
Piezo_Layer = Piezometres_Map + "Piezo.lyrx"
arcpy.SaveToLayerFile_management(in_layer=Output_Layer, out_layer=Piezo_Layer,
is_relative_path="", version="CURRENT")

# Create Basic Map
print("------------------------------------------------------------------------------")
print("Creation of the basic map")
# Provide link to existing project
aprx= arcpy.mp.ArcGISProject(Piezometres_Map + "Piezometres_Map.aprx")

# List the maps in the project
m=aprx.listMaps("Map")[0]
m2=aprx.listMaps("Geneve")[0]


# Define the layer file and add it to the map
lyrFile= arcpy.mp.LayerFile(Piezo_Layer)
m.addLayer(lyrFile,"TOP")
lyr=m.listLayers()[0]
if lyr.name == "Piezo":
    lyr.name = "Piézomètres"

# Define Symbology
sym = lyr.symbology
sym.updateRenderer('SimpleRenderer')
if sym.renderer.type == 'SimpleRenderer':
    sym.renderer.symbol.size = 15
    sym.renderer.symbol.color = {'RGB' : [30, 150, 203, 100]}
    lyr.symbology = sym

aprx.save()
```

Fonction pour la génération des cartes individuelles

La fonction create_map permet de générer la carte de localisation pour un seul piézomètre sélectionné. Le script 7 représente l'implémentation par code. Cette fonction sera intégrée par la suite à une boucle, permettant de générer les rapports pour chaque piézomètre.

Afin de pouvoir mettre en évidence le piézomètre sélectionné, une sélection est réalisée basée sur l'ID_EQUIPEMENT et un nouveau layer file est créé et enregistré (Piezo_Selection2). Cette couche contenant uniquement le piézomètre sélectionné est rajoutée à la carte « Map » ainsi qu'à la carte « Geneve » permettant d'illustrer la position du piézomètre à l'échelle cantonale également. La symbologie est modifiée afin de le faire apparaître en rouge et avec un symbole plus grand que les autres piézomètres dans « Map ». L'étendue de la vue ainsi que le zoom sur la couche de représentation de l'emplacement rapproché (« Map ») sont déterminés à l'aide des fonctions cameras appliquées aux « mapframes ». Ensuite, la carte est exportée en format PNG à un emplacement choisi sur le serveur, portant comme nom « map » avec l'ID_EQUIPEMENT respectif.

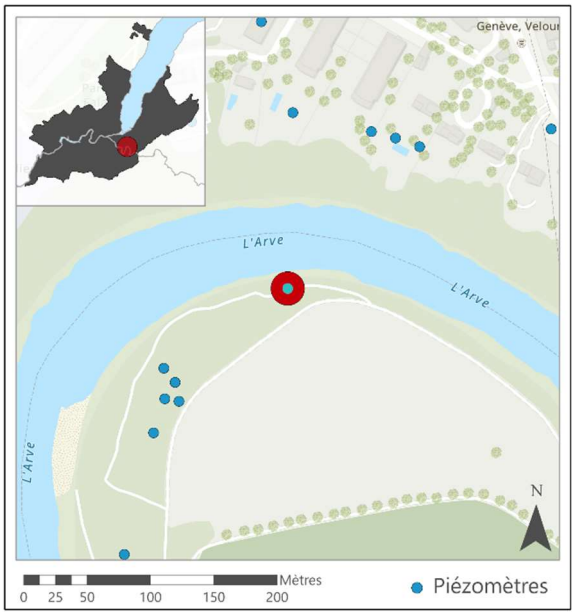


Figure 14: Exemple d'une carte de localisation exportée pour un piézomètre sélectionné, visualisé par le point en rouge. La carte "Geneve" est représentée en miniature dans le coin supérieur gauche et la carte "Map" en plus grand représente l'emplacement rapproché du piézomètre sélectionné avec le zoom et l'étendue précisée dans le script. Les autres piézomètres (Piezo_Layer) rajoutés pendant la première étape de préparation de la carte générale apparaissent en bleu.

Afin de pouvoir recommencer la boucle, la sélection ayant permis de créer la couche avec le piézomètre individuel sélectionné auparavant (Piezo_Selection2) est effacée, la couche est retirée des cartes dans le projet Arc GIS Pro et est également effacée. Le projet est à nouveau enregistré et la fonction pour la génération des cartes individuelles peut être réeffectuée pour un autre piézomètre.

Script 7: Fonction pour la création de carte de localisation pour un piézomètre spécifique (sélection) dans le script "CreationFichePiezo_Serveur.py". Ceci permettra d'automatiser la création de ces cartes pour l'inclusion dans la fiche.

```
# Define function to create Map
print("--------------------------------------------------------------------------------")
print ("Definition of the specific map function to locate individual piezometres")

def create_map(ID_EQUIPEMENT):
    # CREATE MAP

    # Process: Select Layer By Attribute (Select Layer By Attribute)
    criteria = "ID_EQUIPEMENT = " +  str(ID_EQUIPEMENT)
    arcpy.SelectLayerByAttribute_management(in_layer_or_view=lyr,
    selection_type="NEW_SELECTION", where_clause=criteria, invert_where_clause="")

    # Export Selection to a new Layer
    Piezo_Selection1 = Piezometres_Map + "PiezoSelection1.lyrx"
    arcpy.MakeFeatureLayer_management(in_features=lyr, out_layer=Piezo_Selection1,
    where_clause=criteria, workspace="",field_info="OBJECTID OBJECTID VISIBLE
    NONE;SHAPE SHAPE VISIBLE NONE;ID_EQUIPEMENT ID_EQUIPEMENT VISIBLE
    NONE;ID_SONDAGE ID_SONDAGE VISIBLE NONE")
    Piezo_Selection2 = Piezometres_Map + "PiezoSelection2.lyrx"
    arcpy.SaveToLayerFile_management(in_layer=Piezo_Selection1,
    out_layer=Piezo_Selection2, is_relative_path="", version="CURRENT")

    selectionFile= arcpy.mp.LayerFile(Piezo_Selection2)
    m.addLayer(selectionFile,"TOP")
    lyr2=m.listLayers("*PiezoSelection*")[0]

    # Define Symbology for the selected piezometre
    sym2 = lyr2.symbology
    sym2.updateRenderer('SimpleRenderer')
    if sym2.renderer.type == 'SimpleRenderer':
        symbolList2 = sym2.renderer.symbol.listSymbolsFromGallery('')
        symbol2 = symbolList2[55]
        sym2.renderer.symbol = symbol2
        sym2.renderer.symbol.size = 50
        sym2.renderer.symbol.color = {'RGB' : [200, 0, 10, 100]}
        lyr2.symbology = sym2

    #Define Symbology for the Overview Map (Geneva)
    m2.addLayer(selectionFile,"TOP")
    lyr3=m2.listLayers("*PiezoSelection*")[0]

    # Define Symbology for the position of the selected piezometre in the overview map
    sym3 = lyr3.symbology
    sym3.updateRenderer('SimpleRenderer')
    if sym3.renderer.type == 'SimpleRenderer':
        symbolList3 = sym3.renderer.symbol.listSymbolsFromGallery('')
        symbol3 = symbolList3[55]
        sym3.renderer.symbol = symbol3
        sym3.renderer.symbol.size = 30

        sym3.renderer.symbol.color = {'RGB' : [200, 0, 10, 70]}
        lyr3.symbology = sym3

    # List the layouts in the project
    lyt= aprx.listLayouts("Layout2")[0]

    # Define the mapframes in the layout
    mf= lyt.listElements("MAPFRAME_ELEMENT", "Map*")[0]
    mf.camera.setExtent(mf.getLayerExtent(lyr, True, True))
    mf.camera.scale=1500

    #Export Map
    outputname = maps + "map" + str(ID_EQUIPEMENT) + ".png"
    lyt.exportToPNG(outputname, 150)

    # Clear Selection
    arcpy.SelectLayerByAttribute_management(in_layer_or_view=lyr,
    selection_type="CLEAR_SELECTION")

    # Remove Layer (so a new one will be added for the next utilisation)

    m.removeLayer(lyr2)
    m2.removeLayer(lyr3)
    arcpy.Delete_management(Piezo_Selection2)
    # Save the project
    aprx.save()
```

DÉFINITION DES FONCTIONS

Des fonctions, organisées par thématique, ont été définies afin de permettre d'intégrer les informations à une fiche et de faciliter l'exécution automatique des opérations. Ce chapitre s'intéresse donc à la définition de ces fonctions dans le script.

De manière générale, la fiche blanche est appelée « canvas » dans le script. Chaque élément doit être individuellement rajouté au script. Pour intégrer du texte, voire écrire sur la fiche, il est nécessaire de préciser l'emplacement précis avec les coordonnées (x,y) et le texte à rajouter, par exemple canvas.drawString(x, y, « texte à rajouter »). Les caractéristiques concernant le format, comme la police, la taille de ou la couleur, doivent être définis avant (canvas.setFont, canvas.setStrokeColorRGB, canvas.setFillColorRGB). L'interligne est défini dans le code, en adaptant la coordonnée y (p.ex. y-=15). Les illustrations peuvent être rajoutées de la même manière, avec la commande canvas.drawImage (nom du fichier, x, y, largeur de l'image). Ces caractéristiques d'utilisation du module reportLab permettent une grande flexibilité dans la mise en page, mais ont comme désavantage de rapidement rallonger le code. Les commandes reportLab sont utilisées dans les opérations des différentes fonctions thématique pour intégrer l'information respective à la fiche.

Une des difficultés principales était de paramétrer le script afin d'assurer un bon rendu sur la fiche. De très nombreux essais ont donc été requis afin de détecter tous les cas de figures potentiellement problématiques. Ceci inclut le fait de ne pas avoir d'information pour la variable, d'avoir des informations très longues qui recouvraient d'autres attributs ou dépassaient le bord de la feuille ainsi que des problèmes de format (décimales, dates etc.). Afin de mieux représenter les résultats, la mise en page pour chaque fiche est également adaptée en fonction du nombre d'informations (notamment lignes de texte) disponible. De nombreux paramètres supplémentaires ont donc dû être incorporés aux fonctions, afin d'assurer que la fiche d'identité était bien lisible et qu'aucune erreur ne se produisait lors de l'exécution du script.

Avant la définition des fonctions, il est nécessaire de définir les colonnes du tableau synthétiques Excel « Piezo_Copy » qui seront utilisées par la suite. Pour que le code puisse récupérer les données, il est nécessaire de désigner l'onglet dans le tableau Excel dans lequel elles se trouvent (fc = outputname + « \\ »Piezo_Copy$ »). De plus, une liste des piézomètres (« piezos ») est créée, triant ces derniers du plus petit au plus grand en fonction de l'ID_EQUIPEMENT. Cette liste sera utilisée pour la suite afin de pouvoir itérer sur l'ensemble des piézomètres.

Les opérations définies au sein des différentes fonctions sont similaires. Pour cette raison-là, il a été décidé de ne pas inclure d'extraits de codes pour toutes les fonctions, mais uniquement quelques exemples. Le script complet peut être consulté dans les annexes.

Positions de placement

La fonction define_positions place des points de repères pour les différents éléments qui seront intégrés à la fiche. Au sein de cette fonction, les repères horizontaux (x) ainsi que verticaux (y) sont définis, ainsi que les positions d'emplacement des illustrations. Les variables définies dans les fonctions sont à priori uniquement utilisables au sein de la fonction. Afin de permettre la définition dans l'environnement en-dehors de cette fonction, il est nécessaire de préciser qu'il s'agit d'une variable globale (p.ex. global x1).

Titre

La fonction title_function permet de créer l'en-tête de la fiche. Ce dernier définit les emplacements précis et l'allure du logo de l'Etat de Genève, du titre de la fiche, des identifiants généraux utilisés (ID_EQUIPEMENT et ID_SONDAGE) et si disponible, du nom du piézomètre ainsi que de l'identifiant du piézomètre pour le suivi par les sites pollués (OSite).

Script 8: Définition de la fonction title_fonction dans le script "CreationFichePiezo_Serveur.py". Ceci représente un exemple d'une fonction, qui permet de rajouter et de placer l'information sur la feuille blanche.

```
#Define Function to create Title
print("-------------------------------------------------------------------------------")
print("Define functions:")
print ("- title")
def title_function(y):
    #Set Box around title
    canvas.setLineWidth(0.3)
    canvas.setStrokeColorRGB(0.5, 0.5, 0.5, alpha =1)
    canvas.setFillColorRGB(0.2, 0.65, 0.9, alpha = 0.2)
    # Draw a rectangle

    canvas.rect(15, 780, 565, 50, stroke = 1, fill = 1)
    canvas.setFillColorRGB(1, 1, 1, alpha= 1)

    # Set Title
    canvas.setFont('Helvetica-Bold', 14)
    canvas.setFillColorRGB(0, 0, 0)
    canvas.drawString(110,y, "Piézomètre ID_EQUIPEMENT " + ID_EQUIPEMENT)
    canvas.drawString(x11,810, "Sondage " + ID_SONDAGE)
    y-=20
    canvas.setFont('Helvetica', 12)
    if row[2] is not None:
        canvas.drawString(110,y, NOM_PIEZO_TEMP)
    if row[3] is not None:
        canvas.drawString(x11, y, "OSite " + ID_TEMP_SP)
    y-=30

    # Add Logo
    canvas.drawImage(logo, 30, 634, width=58, preserveAspectRatio=True, mask = 'auto')
```

Localisation

La fonction localisation_function organise l'insertion des informations liées à l'emplacement du piézomètre, notamment des coordonnées GPS et de l'adresse du piézomètre.

Accès

La fonction access_fonction insère les informations relatives à l'accès au piézomètre. Ceci inclut la distance à faire à pied pour s'y rendre, les conditions d'accès ainsi que des détails sur l'ouverture du piézomètre (p.ex. si une clef spécifique est requise pour ouvrir la protection)

Activité

La fonction activity_function permet d'intégrer les caractéristiques liées à l'activité du piézomètre, notamment l'état de l'ouvrage, la date de réalisation du sondage et le type d'usage qui en est fait. Pour les piézomètres suivis par le GESDEC, le nom de la tournée ainsi que le contact est également mentionné.

Caractéristiques techniques

La fonction caract_techn_function se concentre sur les caractéristiques techniques du piézomètre. Ceci comprend notamment la profondeur de l'équipement, la profondeur du sondage, la nature du référentiel et la date du dernier nivellement, des informations sur le matériel du tube, la hauteur et la base de la crépine et si le piézomètre est équipé d'une sonde enregistreuse.

Parmi les caractéristiques techniques, les informations qui sont à intégrer sur le schéma du piézomètre disposent d'une fonction à part : schema_function. Cette dernière défini l'emplacement des informations en fonction de l'allure du piézomètre, mais seulement si l'altitude du terrain ainsi que du référentiel sont renseignés. Il existe donc deux différentes options dans le script : celle où l'altitude du référentiel est supérieure à celle du terrain et l'inverse. Ceci permet d'ajuster l'emplacement des informations au schéma approprié. Pour les piézomètres où une des informations concernant l'altitude manque, il n'est pas possible de proposer un schéma. Les informations disponibles sont donc placées en forme de texte à la place du schéma.

Carte de localisation

La fonction create_map a déjà été introduite dans le chapitre précédent. En complément, la fonction addmap_fonction permet de retrouver la carte correspondant au piézomètre sélectionné, basé sur les chemins prédéfinis vers le serveur ainsi que sur l'ID_EQUIPEMENT, et rajoute la carte à la fiche (Image.open, canvas.drawImage).

Photos de piézomètres

La fonction addphotos_function se concentre sur les photos qui doivent être intégrées à la fiche. Elle permet de retrouver les photos correspondantes à l'aide de l'organisation systématique basée sur l'ID_EQUIPEMENT du piézomètre. La prise de vue distante est placée sur la gauche et la prise de vue macro du piézomètre sur la droite. Afin de connaître les coordonnées auxquelles les photos doivent être placées, il est important de connaître l'orientation de la photo (portrait ou paysage). La détermination de l'orientation est effectuée au sein du code, en comparant la hauteur et la largeur de la photo. En fonction de l'orientation, différentes coordonnées d'emplacement sont attribuées. Pour obtenir un meilleur rendu, un cadre correspondant est superposé à la photo.

Script 9: Définition de la fonction addphotos_function dans le script "CreationFichePiezo_Serveur.py"

```
print ("- joining photos")
def addphotos_function():
    # Select Pictures
    global photoList1
    global photoList2
    global heightP1
    arcpy.env.workspace = photos
    criteria1= "*P" + ID_EQUIPEMENT + "_*1.jpg"
    photoList1=arcpy.ListFiles(criteria1)
    criteria2= "*P" + ID_EQUIPEMENT + "_*2.jpg"

    photoList2=arcpy.ListFiles(criteria2)


    # Photo 1
    if photoList1: #should be met when list is not empty
        photo1 = photos + "\\" + photoList1[0]
        PO1 = Image.open(photo1)
        # Check the orientation
        photo1W=PO1.width
        photo1H=PO1.height
        if photo1W > photo1H:
            orientation1 = "Landscape"
            widthP1 = widthH
            yR = yPL
        else:
            orientation1 = "Portrait"
            widthP1 = widthH -50
            yR = yPP
        canvas.drawImage(photo1, xP1, yP, width=widthP1, preserveAspectRatio=True)
        heightP1=photo1H * widthP1/photo1W +1
        canvas.setLineWidth(0.3)
        canvas.setStrokeColorRGB(0.5, 0.5, 0.5, alpha =0.5)
        canvas.rect(xP1, yR, widthP1, heightP1, stroke = 1, fill=0)

    # Photo 2
    if photoList2:
        photo2 = photos + "\\" + photoList2[0]
        PO2= Image.open(photo2)
        photo2W=PO2.width
        #Check the orientation
        photo2H=PO2.height
        if photo2W > photo2H:
            orientation2 = "Landscape"
            widthP2 = widthH
            xP2= xP2L
            yR = yPL
        else:
            orientation2 = "Portrait"
            widthP2=widthH-50
            xP2 = xP2P
            yR = yPP
        canvas.drawImage(photo2, xP2, yP, width=widthP2, preserveAspectRatio=True)
        heightP2=photo2H * widthP2/photo2W
        canvas.setLineWidth(0.3)
        canvas.setStrokeColorRGB(0.5, 0.5, 0.5, alpha =0.5)
        canvas.rect(xP2, yR, widthP2, heightP2, stroke=1, fill=0)
```

Mesures

La fonction measures_function est destinée à intégrer un résumé des mesures à la fiche. Le résumé contient le dernier niveau de nappe mesuré, le niveau de nappe minimum, maximum et moyen ainsi que la température moyenne. De plus, les nombres de mesures effectuées pour les niveaux d'eau et pour les températures sont rajoutés. Le traitement des dates représentait une certaine difficulté, étant donné qu'elles demandaient un traitement particulier dans Python afin de bien s'afficher. Par exemple, le chiffre zéro a dû être manuellement rajouté devant le chiffre, si la valeur du jour ou du mois était inférieur à 10. Ceci était nécessaire afin de permettre un affichage homogène du type JJ.MM.AAAA.

Ressources

La fonction ressources_function rajoute les informations liées aux ressources hydrologiques, notamment le nom de la nappe, son type, la pression ainsi que les unités géologiques atteintes. Pour les piézomètres équipés d'une sonde automatique suivie par la Veille Hydrologique, un lien direct vers le site internet montrant les mesures de ce piézomètre en temps réel est rajouté.

Relevé géologique

La fonction releve_geologique_function est destinée à définir le lien direct vers le relevé géologique du sondage. La grande majorité des relevés est public et les relevés sont disponibles sur internet. Le lien d'accès direct est composé par un lien prédéfini et l'ID_SONDAGE et est donc recomposé au sein du script, pour permettre un accès direct depuis chaque fiche d'identité. Pour les relevés pour lesquels l'accès au log est privé, le chemin d'accès sur le serveur est indiqué à la place.

Aspects visuels de la mise en page

Plusieurs fonctions servent à gérer l'aspect visuel de la mise en page de la fiche, notamment les lignes et les cases colorées. Ces fonctions sont souvent incorporées aux fonctions présentées au-dessus et permettent donc d'ajuster ces paramètres visuels simultanément avec l'ajout du texte. La fonction neatline pose un cadre qui suit les contours de la page. Les fonctions layout_boxes créent des rectangles colorés, qui permettent de mettre en avant certains blocs d'information. La fonction update permet de rajouter la date de création de la fiche à la fin de chaque feuille, en faisant référence à la date du jour.

BOUCLE : CRÉATION DE FICHES D'IDENTITÉ

En vue de générer une fiche PDF pour chaque piézomètre, une boucle est utilisée permettant d'itérer sur la liste de l'ensemble des piézomètres (« for i in piezos »). Le code figure dans le script 10. Toutes les opérations incluses dans la boucle sont reproduites pour chaque élément présent dans la liste, permettant ainsi de créer la fiche d'identité pour chacun des piézomètres.

Dans la boucle, la définition de l'expression précise est requise pour identifier le piézomètre qui sera recherché dans le tableau (« fc » correspond au lien vers l'onglet dans le tableau Excel Piezo_Copy). A l'aide de la fonction arcpy.da.SearchCursor, qui permet de naviguer dans une table en mode lecture, les attributs correspondants au piézomètre sélectionné sont extraits du tableau et définis en tant que variables dans le code. Pour certaines variables, une modification du format est réalisée dès la définition de la variable (p.ex. string, integer). Les variables, pour lesquelles une modification de la forme ou du type du contenu était nécessaire, étaient reprises dans le format brut. Ceci s'explique par la nécessité de devoir intégrer une condition dans le code, qui permet de vérifier si la variable est renseignée, avant de pouvoir adapter le format. Cette étape de vérification est implémentée dans le code dans les fonctions respectives.

Ensuite, les librairies sont à nouveau importées. Cette partie du code a dû être rajoutée, parce qu'il s'est avéré que l'importation des librairies reportLab était requise pour chaque nouvelle création de fiche, à défaut de créer un message d'erreur que la librairie était inconnue.

Comme première étape, le nom du fichier output est défini. Il se compose de « FichePiezo_ » avec l'identifiant du sondage ainsi que « _P » plus l'identifiant d'équipement. Il est enregistré dans le dossier output définit au début du script. Cette composition du nom permet une recherche ciblée du piézomètre par les deux identifiants qui sont fréquemment utilisés au sein du service.

De plus, la taille et l'orientation du « canvas » est définie à cette étape. Ensuite, les fonctions introduites auparavant sont exécutées. A travers ces fonctions, les emplacements sur la fiche sont définis et l'en-tête, le cadre autour de la fiche ainsi que la carte de localisation, le schéma et les photos sont importés. De même, les informations en forme de texte sont rajoutées à travers les fonctions respectives (localisation, accès, activité, caractéristiques techniques).

La difficulté particulière de cette étape est notamment l'organisation de la fiche en fonction des informations disponibles pour chaque piézomètre. Le script doit inclure des conditions permettant d'adapter la mise en page à toutes les configurations éventuelles. Ainsi, deux

différentes options de script ont été élaborées en fonction de la présence ou l'absence de photos.

Si aucune photo n'existe, les fonctions permettant de rajouter les informations sur les mesures, ressources et relevé géologique sont exécutées afin de rajouter l'information sur la même page. Le PDF est enregistré. En revanche, si des photos existent, ces dernières occupent la première page et il est donc nécessaire de rajouter une deuxième page à la fiche. Etant donné que cette nouvelle page est vide, il est nécessaire de réexécuter les fonctions de mise en page (création de l'en-tête, cadre et définition des emplacements). Il reste à préciser que les emplacements précis des blocs d'informations varient également si une deuxième page est ajoutée, ce qui a dû être paramétré dans le script. Afin de rajouter les informations sur cette deuxième page, les fonctions des mesures, des ressources et du relevé géologique sont exécutées et le PDF est enregistré.

Ainsi, une fiche d'identité adaptée aux informations disponibles pour chaque piézomètre peut être obtenue de manière automatique pour l'ensemble des plus de 5000 piézomètres de la base de données.

Script 10: Boucle qui permet d'itérer sur la liste de l'ensemble des piézomètres et de générer les fiches d'identité individuelles

```
### LOOP THROUGH ALL EXISTING PIEZOMETRES TO CREATE REPORT
print("-------------------------------------------------------------------------")
print("Creation of the reports in loop:")
print("-------------------------------------------------------------------------")
for i in piezos:
    expression = '"ID_EQUIPEMENT" = ' + str(i)
    print("For piezometre: " + expression)
    # Grasp the data for each piezometre
    print("- extraction of the individual data")
    with arcpy.da.SearchCursor(fc, fields, expression) as cursor:
        for row in cursor:
            ID_EQUIPEMENT = str(int(float(row[0])))
            ID_SONDAGE = str(int(float(row[1])))
            NOM_PIEZO_TEMP =str(row[2])
            ID_TEMP_SP = str(row[3])
            COORD_X =str(row[4])
            COORD_Y=str(row[5])
            RUE=str(row[6])
            NPA=row[7]
            COMMUNE=str(row[8])
            DISTANCE_A_PIEDS=str(row[9])
            CONDITION_ACCES=str(row[10])
            OUVERTURE=str(row[11])
            ETAT_EQU = str(row[12])
            USAGE = str(row[13])
            TOURNEE= str(row[14])
            FIN_TRAVAUX = row[15]          #no string possible as formatting
            required for the places after the comma
            DIA_FOND_TROU = row[16]
            ALT_TERRAIN = row[17]          #no string possible as formatting
            required for the places after the comma
            ALT_REFERENTIEL = row[18]      #no string possible as formatting
            required for the places after the comma
            REFERENTIEL = str(row[19])
            DATE_NIVELLEMENT = str(row[20])
            PROFONDEUR_EQUIP = row[21]     #no string possible as formatting
            required for the places after the comma
            PROFONDEUR_MD = row[22]        #no string possible as formatting
            required for the places after the comma
            TUBAGE = str(row[23])
            DIA_TUBE= row[24]              #no string possible as formatting
            required for the places after the comma
            H_CREPINE = row[25]
            B_CREPINE=row[26]
            SONDE_ENREGISTREUSE = str(row[27])
            NOM = str(row[28])
            NAPPE = str(row[29])
            PRESSION_NAPPE = str(row[30])
            UNITE_GEOL_1 = str(row[31])
            UNITE_GEOL_2 = str(row[32])
            AUTOMATIQUE =str(row[33])
            LIEN_VH = str(row[34])
            COUNT_NIVEAU_MESURE = row[35]
            MEAN_NIVEAU_NAPPE = row[36]    #no string possible as formatting
            required for the places after the comma
            MIN_NIVEAU_NAPPE = row[37]
            MAX_NIVEAU_NAPPE = row[38]
            MEAN_TEMPERATURE = row[39]
            MAX_DATE_MESURE = row[40]
            MIN_Date = row[41]
            MAX_Date = row[42]
            DER_NIVEAU_NAPPE = row[43]
            COUNT_TEMPERATURE=row[44]

    ### IMPORT LIBRARIES
    # Import reportlab
    from reportlab.pdfgen import canvas
    from reportlab.lib.pagesizes import A4, portrait
    from reportlab.lib import colors
    from reportlab import platypus
    from reportlab.lib.styles import ParagraphStyle as PS
    from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph

    #Import PIL
    from PIL import Image

    #Import date
    from datetime import date

    ### DEFINE THE CANVAS COMPOSITION
    print("- definition of the canvas caracteristics")
    # Define name of the canvas
    pdfname = outputpath + "FichePiezo_" +str(ID_SONDAGE)+ "_P" + str(ID_EQUIPEMENT)
    + ".pdf"

    # Define canvas orientation
    canvas=canvas.Canvas(pdfname,pagesize=portrait(A4))
    canvas.width, canvas.height = A4

    # Define positions
    define_positions()

    # Define Title
    title_function(y)

    # Define Neatline
    neatline()

    # Add MAPS
    print("- creation of the map and inclusion on the canvas")
    create_map(ID_EQUIPEMENT)
    addmap_function()

    # Add Schema
    print("- creation of the schema and inclusion on the canvas")
    schema_function(ALT_TERRAIN, ALT_REFERENTIEL, DIA_TUBE, DIA_FOND_TROU)

    # Localisation
    print("- add information on localisation")
    localisation_function(NPA)

    # Access
    print("- add information on access")
    access_function()

    # Activity
    print("- add information on activity")
    activity_function()

    # Caractéristiques techniques
    print("- add information on technical caracteristics")
    caract_techn_function(PROFONDEUR_EQUIP, PROFONDEUR_MD, REFERENTIEL,
    DATE_NIVELLEMENT, TUBAGE, H_CREPINE, B_CREPINE, SONDE_ENREGISTREUSE)

    # Add PHOTOS
    print("- add photos for this specific piezometre (if available)")
    addphotos_function()

    # Define Layout Boxes for Page 1 (depending on the elements available for this
    piezometre)
    layout_boxesP0()
    if not photoList1 and not photoList2:
        PresencePhoto = 0
    else:
        PresencePhoto = 1
        layout_boxesP2()

    # Insert Resources if no photos are available
    if PresencePhoto==0:
        # ADD INFORMATION ABOUT MEASURES
```

```
            print("- add information on measures")
            y = yR
            yMesureTop= yR + 10
            measures_function(MAX_DATE_MESURE, MIN_Date, MAX_Date, COUNT_NIVEAU_MESURE,
            MEAN_NIVEAU_NAPPE, DER_NIVEAU_NAPPE, COUNT_TEMPERATURE, MEAN_TEMPERATURE)

            # ADD INFORMATION ABOUT RESSOURCES
            print("- add information on resources")
            y = yR
            ressources_function()

            line_function()

            # ADD Link to LOG
            print("- add link to LOG")
            releve_geologique_function()

            #Add Date
            update()

            # Save PDF
            canvas.save()


        # Add another page if photos are available
        elif PresencePhoto == 1: #si photos sont présentes, une deuxième page est rajoutée

            #RAJOUT SECONDE PAGE
            canvas.showPage()
            define_positions()

            #Define Title
            title_function(y)

            # Define Neatline
            neatline()

            # ADD INFORMATION ABOUT MEASURES
            print("- add information on measures")
            y = yS2
            yMesureTop= yS2 + 7
            measures_function(MAX_DATE_MESURE, MIN_Date, MAX_Date, COUNT_NIVEAU_MESURE,
            MEAN_NIVEAU_NAPPE, DER_NIVEAU_NAPPE, COUNT_TEMPERATURE, MEAN_TEMPERATURE)


            # ADD INFORMATION ABOUT RESSOURCES
            print("- add information on resources")
            y=yS2
            ressources_function()

            # ADD Link to LOG
            print("- add link to LOG")
            releve_geologique_function()

            #Add Date
            update()

            #Save PDF
            canvas.save()

        print ("PDF successfully saved")

        del canvas


        print("--------------------------------------------------------------------------------
        ")

    #Delete Layer from the map
    m.removeLayer(lyr)

    #Script finished

    print ("Script has successfully run")
```

FIN DU SCRIPT

Après la finalisation de la boucle et en arrivant à la fin du script, une opération permet de remettre au point de départ les projets ArcGIS Pro utilisés pour ce script. Ainsi, la couche des piézomètres « Piezo_Layer » est retirée de la carte. Ceci permet de remettre les données input à l'état initial et de pouvoir relancer le script à tout autre moment.

Cependant, si le script n'arrive pas à son terme en raison d'une erreur, cette opération n'est pas réalisée. Lors d'une nouvelle exécution du script, une couche additionnelle est donc rajoutée. Ainsi, plusieurs couches « Piezo_Layer » risquent de se superposer et il est donc recommandé de retirer manuellement cette couche si ce cas de figure devait se produire.

DIFFÉRENTES VERSIONS DU SCRIPT

Le temps d'exécution du script pour l'ensemble des plus de 5044 fiches est important (~16h). Afin de proposer une solution viable, un script modifié, qui permet de générer des fiches pour les nouveaux piézomètres uniquement, a été créé. Les autres fiches ne sont cependant pas actualisées. La seule différence comparée au script principal est l'inclusion d'une condition supplémentaire au début de la boucle, à travers laquelle il est possible de vérifier si la fiche pour ce piézomètre existe déjà. Si tel est le cas, le script passe au prochain piézomètre dans la liste, sinon une fiche d'identité est générée.

Script 11: Condition supplémentaire intégrée à la boucle pour l'actualisation hebdomadaire. Cette version du script est intitulée "CreationFichePiezo_Weekly_Serveur.py »

```
#Check whether the file is already existing
file = "_P" + str(ID_EQUIPEMENT) + ".pdf"
if os.path.isfile(outputpath + "FichePiezo_" +str(ID_SONDAGE)+ "_P" + str(ID_EQUIPEMENT) + ".pdf"):
    print("The file exists")
    pass
else:
    print("The file does not exist")
    print("An identity file is being created")
```

3. RÉSULTATS

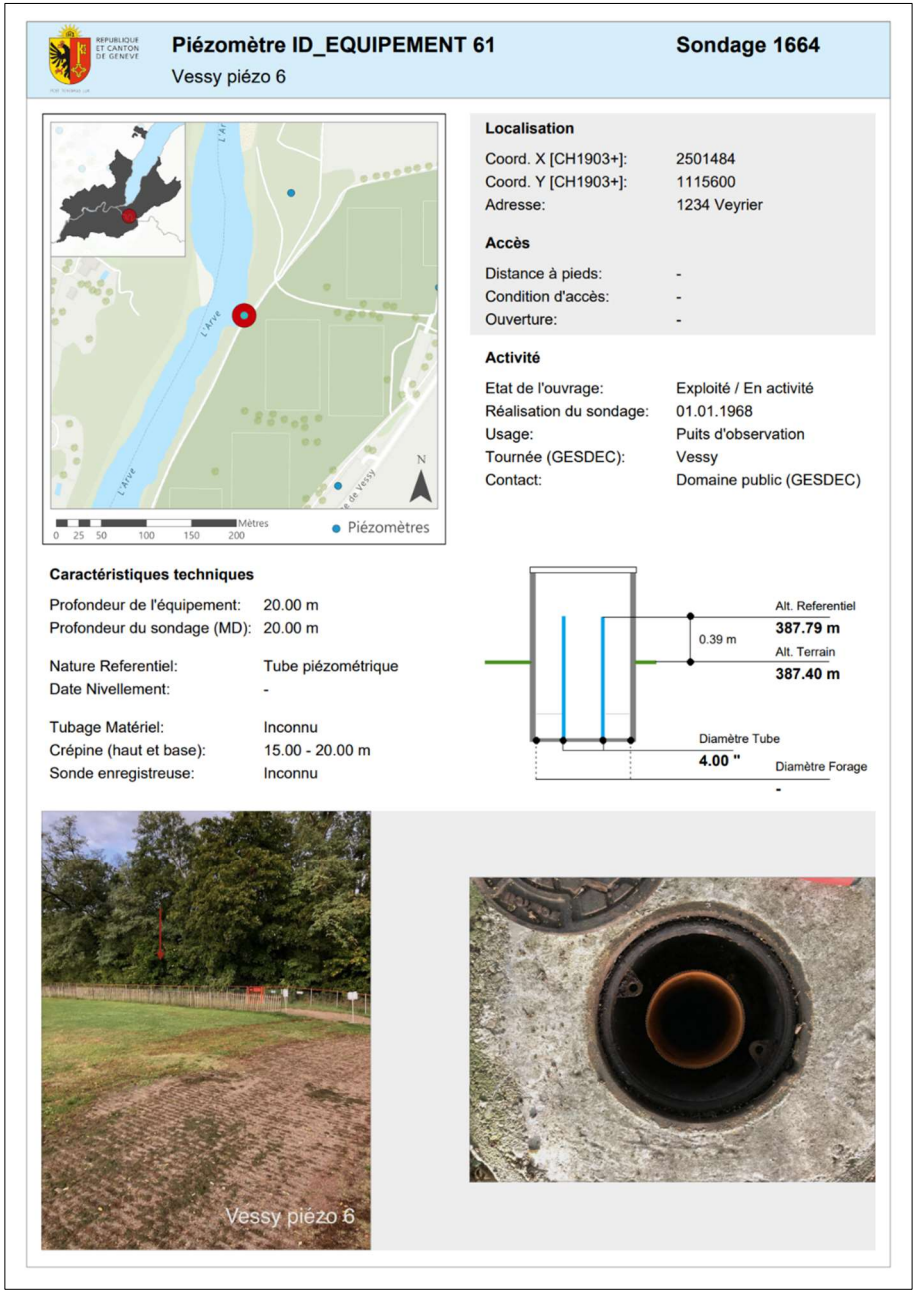
Les fiches d'identités représentent le résultat tangible de ce processus de travail, à côté de la mise à disposition du script permettant de générer les fiches de manière automatique. Les fiches pour l'ensemble des plus de 5000 piézomètres sont directement mises à disposition sur le serveur et donc accessibles pour tout le service, à tout moment.

Afin de donner un aperçu de la réalisation concrète, certains exemples ont été sélectionnés. Les fiches d'identités pour les piézomètres avec l'ID_EQUIPEMENT 61 et 582 représentent des exemples de fiches bien renseignées (cf. figures 15 et 16). Les deux fiches montrent les deux variantes de schémas et affichent les photos respectives. La fiche du piézomètre 63, représentée dans la figure 17, montre la configuration pour une fiche également bien renseignée mais sans photos. Contrairement aux autres exemples, la fiche pour le piézomètre 71 est cependant pauvrement renseignée (cf. figure 18), car peu d'informations sont disponible dans la base de données.

Pour les piézomètres suivis par la veille hydrologique, un lien intégré à la fiche permet de directement accéder aux graphiques (par exemple pour le piézomètre 63, cf. figure 19).

Le relevé géologique est également consultable en ligne pour la majorité des piézomètres et un hyperlien permet d'y accéder directement. La figure 20 montre la représentation graphique du log pour le piézomètre 63 qui est accessible à travers ce lien.

Une grille de lecture pour les différentes informations présentes sur la fiche d'identité est mise à disposition dans le tableau 3. Il fournit des descriptions supplémentaires sur les variables ainsi que leur source précise (Nom de la table attributaire et le ou les attribut(s) précis).



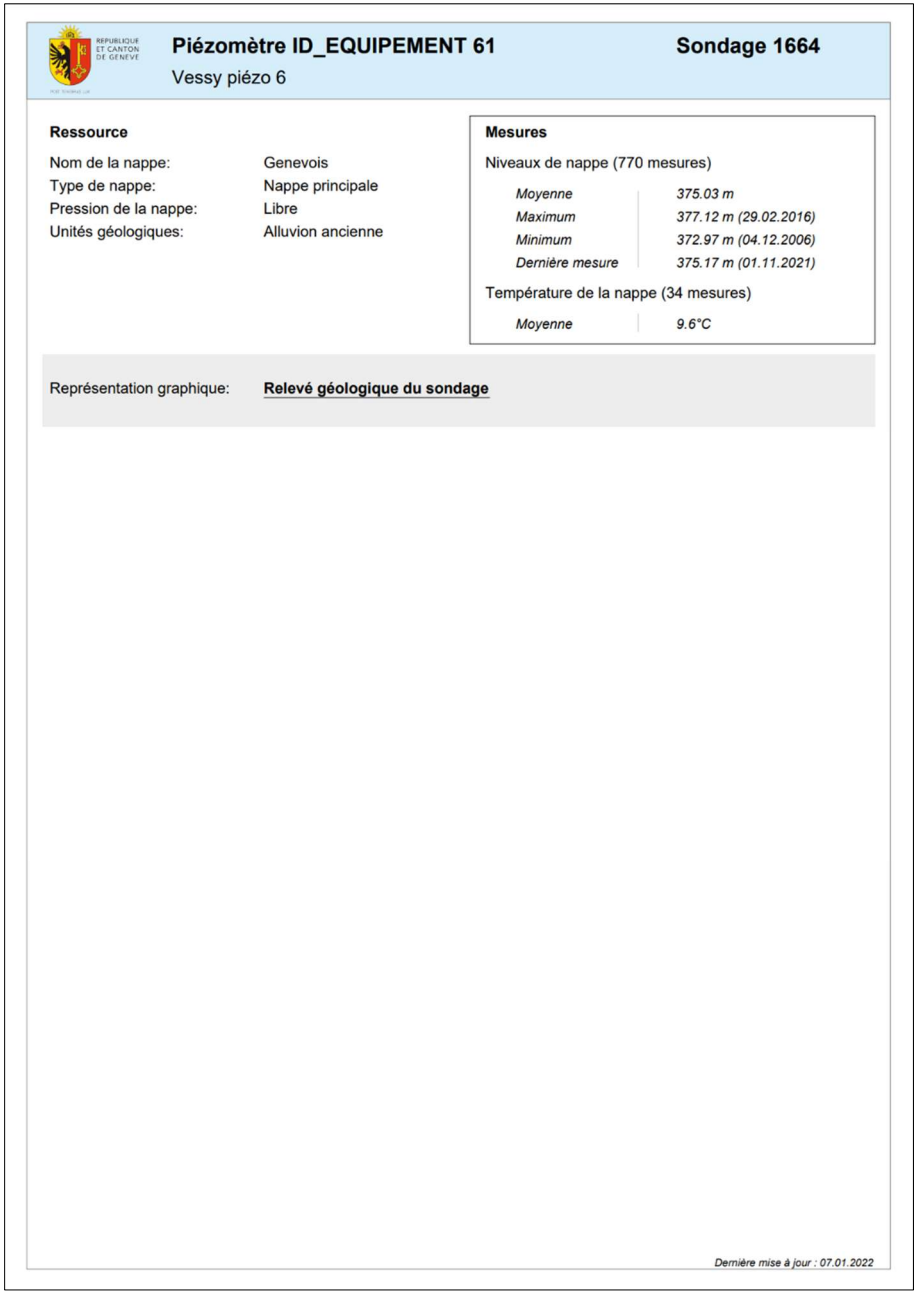
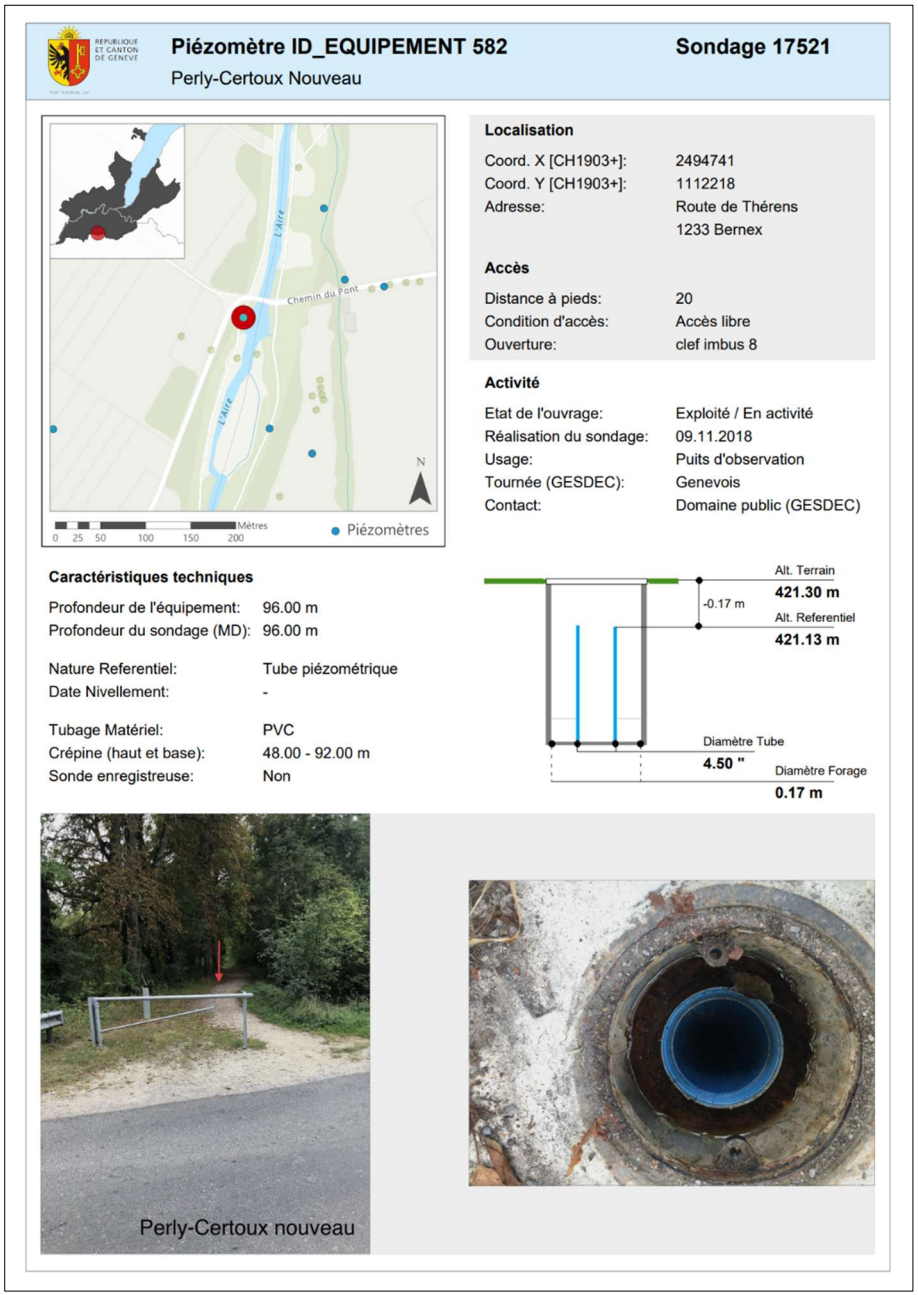


Figure 15: Fiche d'identité pour le piézomètre avec l'ID_EQUIPEMENT 61 (Version PDF disponible dans l'Annexe 2)



Piézomètre ID_EQUIPEMENT 582 — **Sondage 17521**

Perly-Certoux Nouveau

Localisation

Coord. X [CH1903+]:	2494741
Coord. Y [CH1903+]:	1112218
Adresse:	Route de Thérens 1233 Bernex

Accès

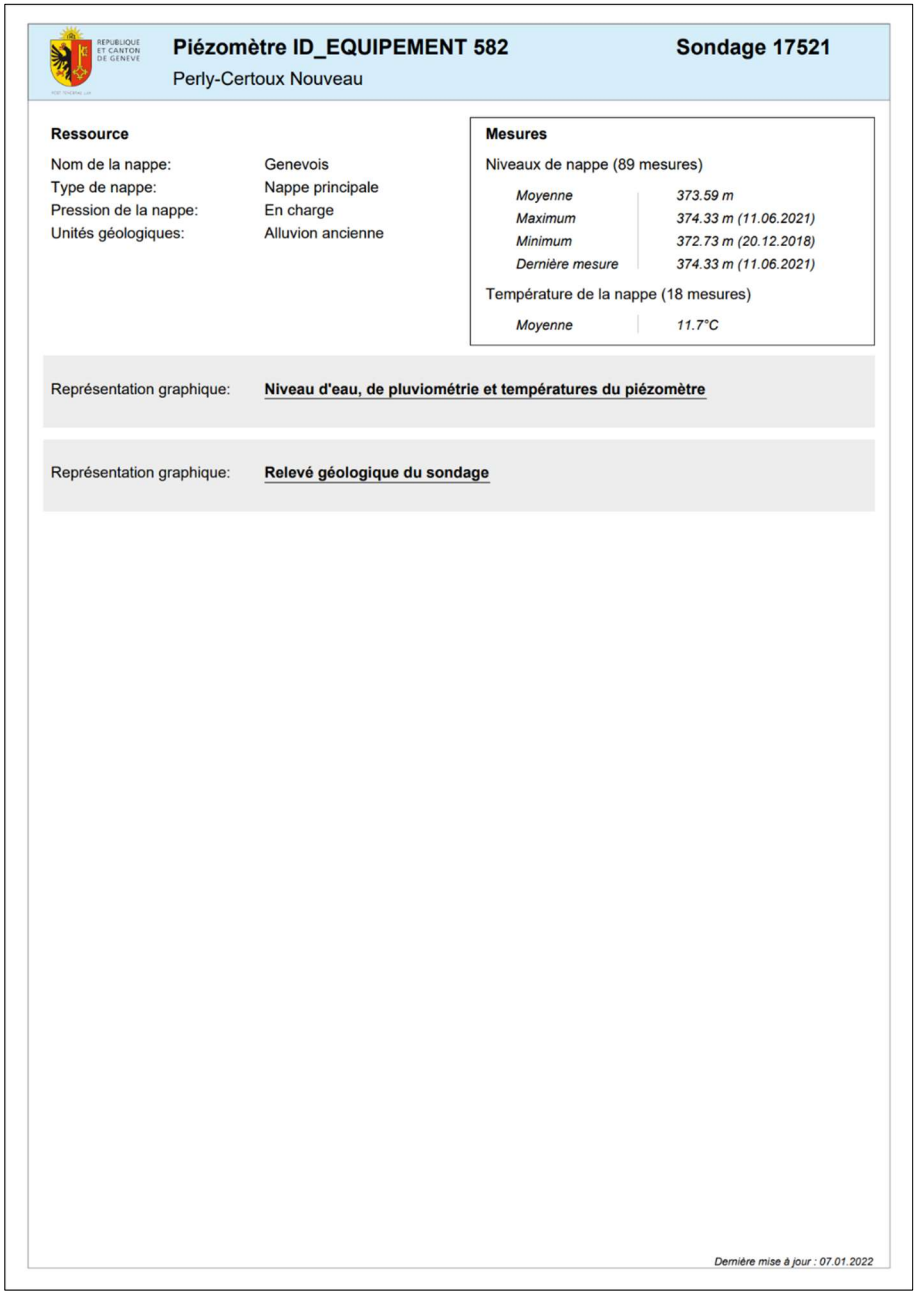
Distance à pieds:	20
Condition d'accès:	Accès libre
Ouverture:	clef imbus 8

Activité

Etat de l'ouvrage:	Exploité / En activité
Réalisation du sondage:	09.11.2018
Usage:	Puits d'observation
Tournée (GESDEC):	Genevois
Contact:	Domaine public (GESDEC)

Caractéristiques techniques

Profondeur de l'équipement:	96.00 m
Profondeur du sondage (MD):	96.00 m
Nature Referentiel:	Tube piézométrique
Date Nivellement:	-
Tubage Matériel:	PVC
Crépine (haut et base):	48.00 - 92.00 m
Sonde enregistreuse:	Non



Piézomètre ID_EQUIPEMENT 582 — **Sondage 17521**

Perly-Certoux Nouveau

Ressource

Nom de la nappe:	Genevois
Type de nappe:	Nappe principale
Pression de la nappe:	En charge
Unités géologiques:	Alluvion ancienne

Mesures

Niveaux de nappe (89 mesures)

Moyenne	*373.59 m*
Maximum	*374.33 m (11.06.2021)*
Minimum	*372.73 m (20.12.2018)*
Dernière mesure	*374.33 m (11.06.2021)*

Température de la nappe (18 mesures)

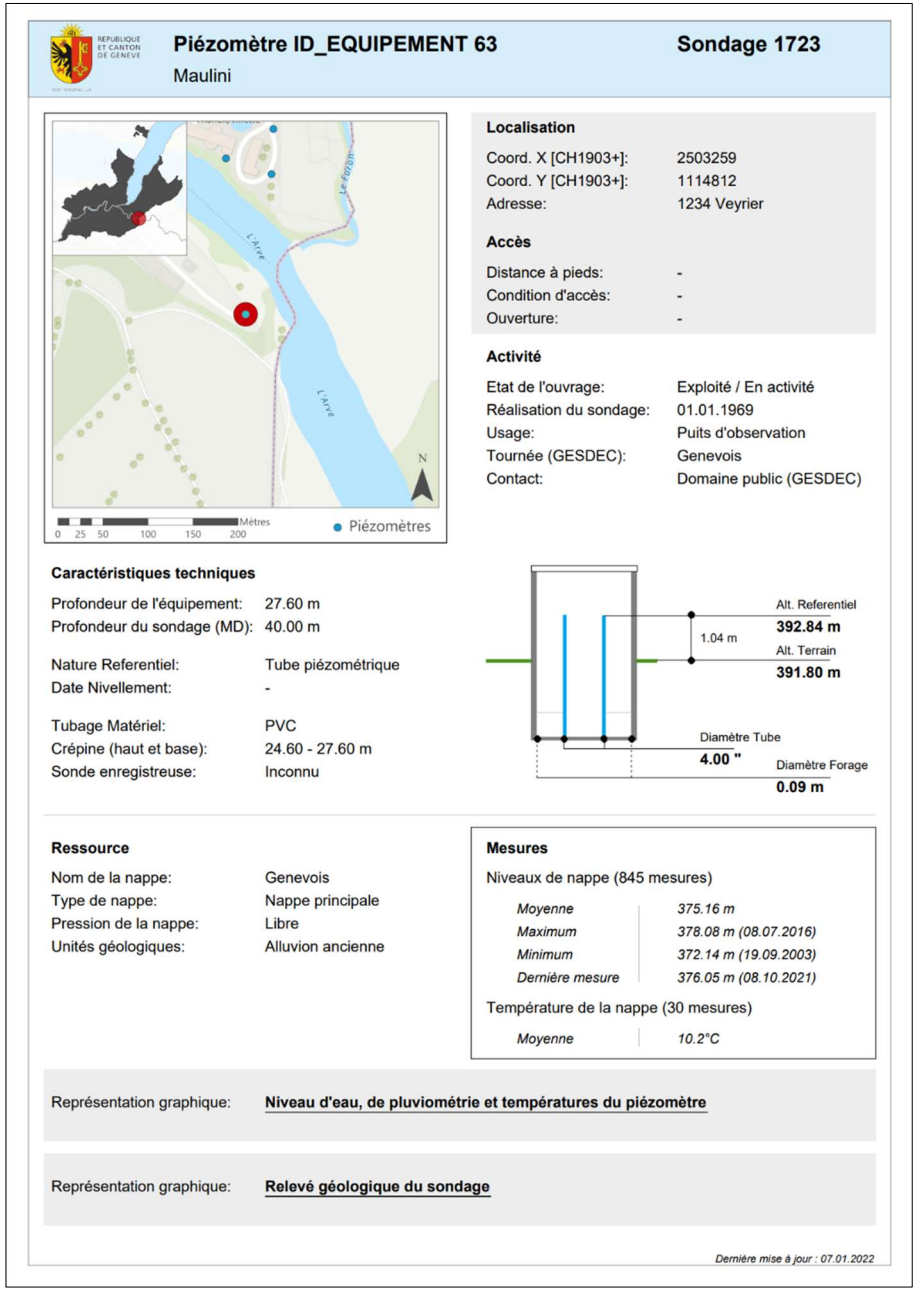
Moyenne	*11.7°C*

Représentation graphique: **Niveau d'eau, de pluviométrie et températures du piézomètre**

Représentation graphique: **Relevé géologique du sondage**

Dernière mise à jour : 07.01.2022

Figure 16: Fiche d'identité pour le piézomètre avec l'ID_EQUIPEMENT 582 (Version PDF disponible dans l'Annexe 2)



Piézomètre ID_EQUIPEMENT 63 — **Sondage 1723**

Maulini

Localisation

Coord. X [CH1903+]:	2503259
Coord. Y [CH1903+]:	1114812
Adresse:	1234 Veyrier

Accès

Distance à pieds:	-
Condition d'accès:	-
Ouverture:	-

Activité

Etat de l'ouvrage:	Exploité / En activité
Réalisation du sondage:	01.01.1969
Usage:	Puits d'observation
Tournée (GESDEC):	Genevois
Contact:	Domaine public (GESDEC)

Caractéristiques techniques

Profondeur de l'équipement:	27.60 m
Profondeur du sondage (MD):	40.00 m
Nature Referentiel:	Tube piézométrique
Date Nivellement:	-
Tubage Matériel:	PVC
Crépine (haut et base):	24.60 - 27.60 m
Sonde enregistreuse:	Inconnu

Ressource

Nom de la nappe:	Genevois
Type de nappe:	Nappe principale
Pression de la nappe:	Libre
Unités géologiques:	Alluvion ancienne

Mesures

Niveaux de nappe (845 mesures)

Moyenne	*375.16 m*
Maximum	*378.08 m (08.07.2016)*
Minimum	*372.14 m (19.09.2003)*
Dernière mesure	*376.05 m (08.10.2021)*

Température de la nappe (30 mesures)

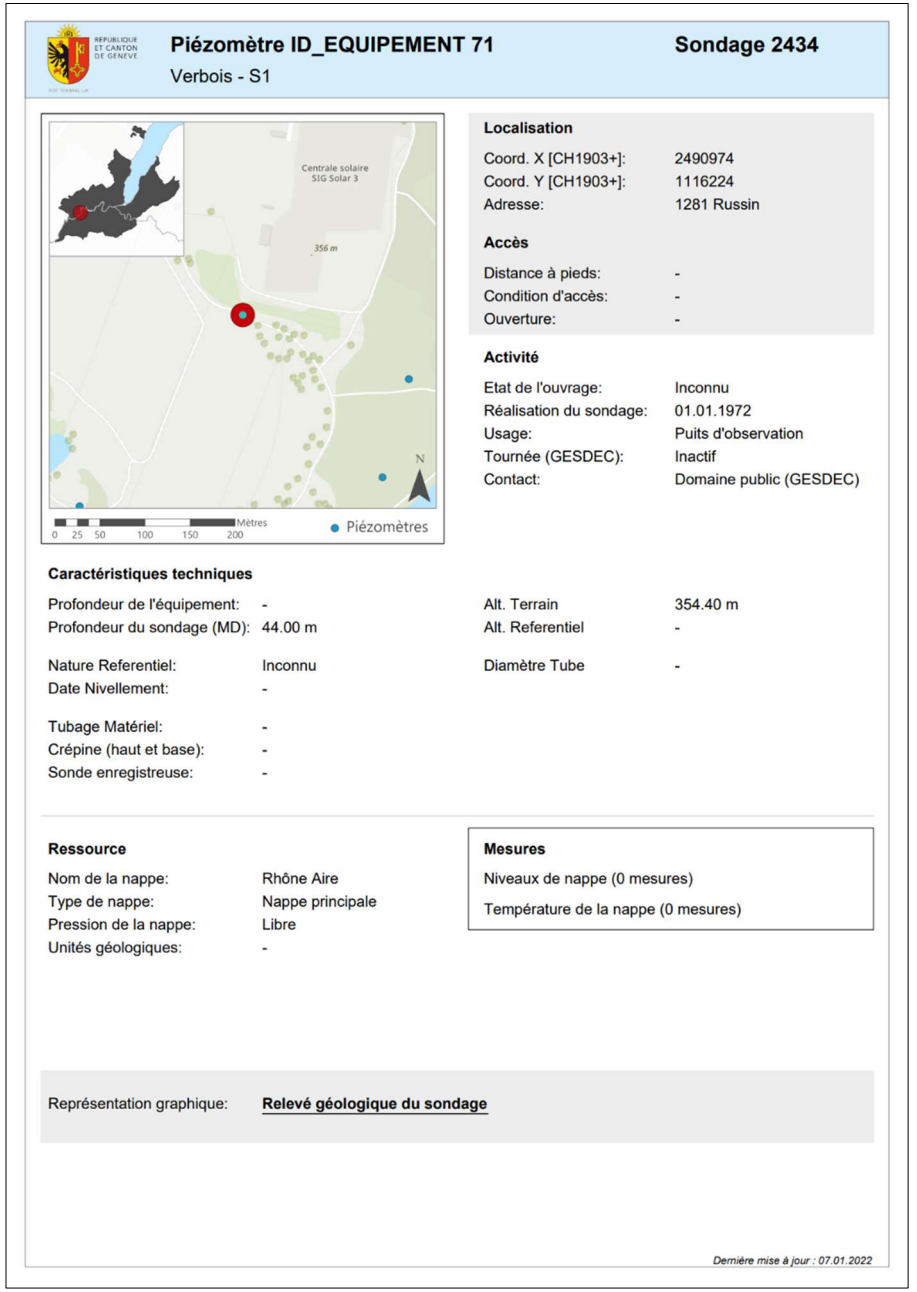
Moyenne	*10.2°C*

Représentation graphique: **Niveau d'eau, de pluviométrie et températures du piézomètre**

Représentation graphique: **Relevé géologique du sondage**

Dernière mise à jour : 07.01.2022

Figure 17: Fiche d'identité pour le piézomètre avec l'ID_EQUIPEMENT 63 (Version PDF disponible dans l'Annexe 2)



Piézomètre ID_EQUIPEMENT 71 — **Sondage 2434**

Verbois - S1

Localisation

Coord. X [CH1903+]:	2490974
Coord. Y [CH1903+]:	1116224
Adresse:	1281 Russin

Accès

Distance à pieds:	-
Condition d'accès:	-
Ouverture:	-

Activité

Etat de l'ouvrage:	Inconnu
Réalisation du sondage:	01.01.1972
Usage:	Puits d'observation
Tournée (GESDEC):	Inactif
Contact:	Domaine public (GESDEC)

Caractéristiques techniques

Profondeur de l'équipement:	-
Profondeur du sondage (MD):	44.00 m
Nature Referentiel:	Inconnu
Date Nivellement:	-
Tubage Matériel:	-
Crépine (haut et base):	-
Sonde enregistreuse:	-
Alt. Terrain	354.40 m
Alt. Referentiel	-
Diamètre Tube	-

Ressource

Nom de la nappe:	Rhône Aire
Type de nappe:	Nappe principale
Pression de la nappe:	Libre
Unités géologiques:	-

Mesures

Niveaux de nappe (0 mesures)

Température de la nappe (0 mesures)

Représentation graphique: **Relevé géologique du sondage**

Dernière mise à jour : 07.01.2022

Figure 18: Fiche d'identité pour le piézomètre avec l'ID_EQUIPEMENT 71 (Version PDF disponible dans l'Annexe 2)

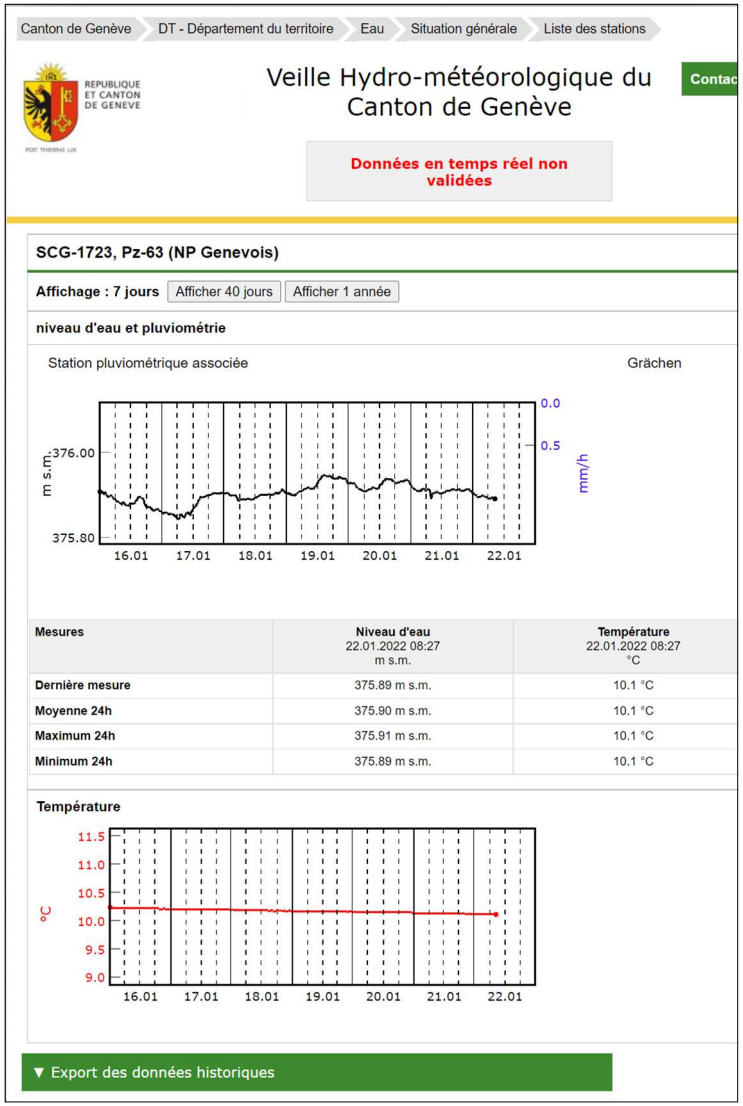


Figure 19: Représentation graphique du niveau d'eau, de pluviométrie et de température pour le piézomètre avec l'ID_EQUIPEMENT 63, directement accessible à travers l'hyperlien intégré à la fiche

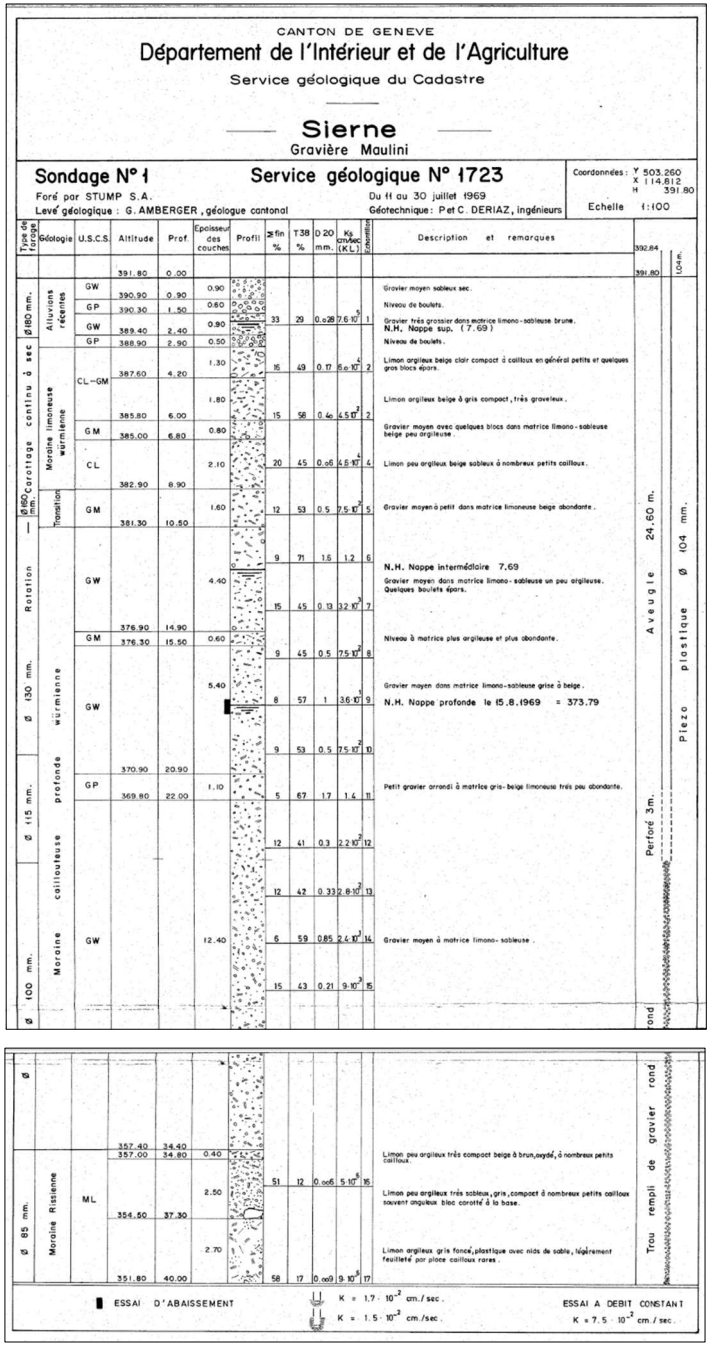


Figure 20 (sur la droite): Représentation graphique du relevé géologique (log) pour le piézomètre avec l'ID_EQUIPEMENT 63, directement accessible à travers l'hyperlien intégré à la fiche d'identité

Tableau 3: Description des informations sélectionnées pour la fiche d'identité ainsi que leurs sources directes. (Source: Créé par l'auteure et inspiré par République et Canton de Genève (2020b)).

Information Fiche	Description	Attribut	Source (Couche SOLSTISS)
Identifiants			
Identifiant équipement	Identifiant d'équipement caractéristique pour le piézomètre en question	ID_EQUIPEMENT	GEOL_SS_EQUIPEMENT
Sondage	Identifiant du sondage	ID_SONDAGE	GEOL_SS_SONDAGE
Nom du piézomètre	Nom du piézomètre utilisé au sein du service	NOM_PIEZO_TEMP	GEOL_SS_EQUIPEMENT
Osite	Identifiant temporaire du piézomètre selon la classification du suivi des sites pollués	ID_temp_SP	GEOL_SS_EQUIPEMENT
Localisation			
Coordonnée X	Coordonnée GPS X de l'emplacement du piézomètre (Système de coordonnées CH1903+)	COORD_X	GEOL_SS_SONDAGE
Coordonnée Y	Coordonnée GPS Y de l'emplacement du piézomètre (Système de coordonnées CH1903+)	COORD_Y	GEOL_SS_SONDAGE
Adresse	Emplacement du piézomètre exprimé en tant qu'adresse (précision de 50m)		A.CAD_ADRESSE A.GMO_ROUTIER A.AGGLO_COMMUNES A.GEO_ADRESSE_AGGLO
Accès			
Distance à pieds	Information sur la distance à parcourir à pieds (depuis la route p.ex.)	DISTANCE_A_PIEDS	GEOL_SS_EQUIPEMENT
Condition d'accès	Information sur les conditions pour accéder au piézomètre (p.ex. verrouillé ou accès libre)	CONDITION_ACCES	GEOL_SS_EQUIPEMENT
Ouverture	Matériel requis pour accéder à l'équipement	OUVERTURE	GEOL_SS_EQUIPEMENT
Activité			
Etat de l'ouvrage	Etat de l'équipement: Inconnu, Exploité/En activité, Exploitable, Défectueux, Cimenté/Rebouché	ETAT_EQU	GEOL_SS_EQUIPEMENT
Date d'installation	Date de fin d'exécution des travaux	FIN_TRAVAUX	GEOL_SS_ADMIN
Usage	Utilisation de l'équipement (p.ex. puits d'observation, géothermie)	USAGE	GEOL_SS_EQUIPEMENT
Tournée (GESDEC)	Renseigne sur la tournée dans laquelle se trouve le piézomètre (à condition d'être suivi par le GESDEC)	TOURNEE	GEOL_SS_EQUIPEMENT
Contact	Coordonnées de contact du gesdec si le piézomètre est suivi par les entités publiques		

Tableau 3 (suite) : Description des informations sélectionnées pour la fiche d'identité ainsi que leurs sources directes. (Source: Créé par l'auteure et inspiré par République et Canton de Genève (2020b)).

Information Fiche	Description	Attribut	Source (Couche SOLSTISS)
Caractéristiques techniques			
Profondeur de l'équipement	Altitude [en m] atteint par l'équipement du sondage	PROFONDEUR_EQUIP	GEOL_SS_EQUIPEMENT
Profondeur du sondage (MD)	Altitude [en m] atteint au fond du sondage	PROFONDEUR_MD	GEOL_SS_SONDAGE
Nature Referentiel	Information sur le référentiel utilisé pour les mesures (p.ex. tube piézométrique, surface du terrain naturel)	REFERENTIEL	GEOL_SS_EQUIPEMENT
Date Nivellement	Renseigne la dernière date de nivellement de l'équipement	DATE_NIVELLEMENT	GEOL_SS_EQUIPEMENT
Tubage Matériel	Indique le type de tubage employé pour ce sondage (p.ex. métallique, PVC)	TUBAGE	GEOL_SS_EQUIPEMENT
Crépine (haut et base)	Indique la profondeur à laquelle se trouvent le sommet et la base de la crépine (tube perforé, qui permet l'infiltration d'eau)	H_CREPINE / B_CREPINE	GEOL_SS_EQUIPEMENT
Sonde automatique	Renseigne si le puit est équipé d'une sonde enregistreuse	SONDE_ENREGISTREUSE	GEOL_SS_EQUIPEMENT
Altitude Terrain	Information sur l'altitude du terrain naturel	ALT_TERRAIN	GEOL_SS_SONDAGE
Altitude Referentiel	Information sur l'altidue de référence utilisées pour effectuer les mesures	ALT_REFERENTIEL	GEOL_SS_EQUIPEMENT
Diamètre Tube	Informations sur le diamètre du tube situé dans le sondage	DIA_TUBE	GEOL_SS_EQUIPEMENT
Diamètre Forage	Informations sur le plus petit diamètre mesuré dans le sondage (permet de savoir quels équipements peuvent être installés)	DIA_FOND_TROU	GEOL_SS_SONDAGE
Ressource			
Nom de la nappe	Renseigne le nom de la ressource hydrogéologique	NOM	GEOL_SS_RESSOURCE
Type de la nappe	Information sur la nature de la nappe identifiée (p.e.x nappe principale, nappe superficielle, nappe profonde)	NAPPE	GEOL_SS_RESSOURCE
Pression de la nappe	Indique des caractéristiques de pression de la nappe (le cas échéant)	PRESSION_NAPPE	GEOL_SS_RESSOURCE
Unités géologiques	Informations sur l'unité géologique primaire et secondaire éventuelle de la roche-réservoir.	UNITE_GEOL_1 / UNITE_GEOL_2	GEOL_SS_RESSOURCE
Mesures			
Nombre de mesures de niveaux d'eau	Indique le nombre de mesures de niveaux d'eau effectué sur ce piézomètre	NIVEAU_NAPPE / DATE_MESURE	GEOL_SS_SONDAGE_MESURE
Moyenne Niveau de nappe	Renseigne le niveau moyen de la nappe (à partir de 3 mesures disponibles)	NIVEAU_NAPPE / DATE_MESURE	GEOL_SS_SONDAGE_MESURE
Maximum	Renseigne la mesure de niveaux d'eau maximum et sa date	NIVEAU_NAPPE / DATE_MESURE	GEOL_SS_SONDAGE_MESURE
Minimum	Renseigne la mesure de niveaux d'eau minimum et sa date	NIVEAU_NAPPE / DATE_MESURE	GEOL_SS_SONDAGE_MESURE
Dernière mesure	Renseigne la dernière mesure de niveaux d'eau effectuée ainsi que sa date	NIVEAU_NAPPE / DATE_MESURE	GEOL_SS_SONDAGE_MESURE
Nombre de mesures de température	Indique le nombre de mesures de niveaux d'eau effectuées sur ce piézomètre	TEMPERATURE/ DATE_MESURE	GEOL_SS_SONDAGE_MESURE
Moyenne de température	Renseigne la moyenne des mesures de température	TEMPERATURE	GEOL_SS_SONDAGE_MESURE

4. DISCUSSION

La discussion permettra de développer les enjeux liés au travail de stage. Dans une première partie, les aspects portant sur le projet de stage seront thématisés et la deuxième partie se concentrera sur le déroulement du stage.

PROJET DE STAGE : CRÉATION AUTOMATISÉE DE FICHES D'IDENTITÉ

Ce travail de stage a permis de mettre en lumière les défis liés à la mise à disposition optimale des géodonnées. SOLSTISS, qui incorpore, stocke et fournit les informations du sous-sol, est un système d'information avec une structure complexe, qui permet de gérer les données au mieux. Cependant, cette organisation complexe de données demande à l'utilisateur·trice de naviguer entre les différentes tables attributaires liées, pour retrouver une information attributaire spécifique. La recherche de photos doit également être effectuée sur le serveur dans différents dossiers. La mise à disposition d'un résumé des informations et illustrations disponibles par piézomètre, dans un format pratique à l'emploi, permet donc d'optimiser l'accès aux données et de faciliter le travail.

Le principal résultat de ce travail consiste en la création du script Python, qui permet la génération automatisée de fiches d'identité pour les piézomètres. Tenant compte de l'important nombre de piézomètres, dépassant les 5000 équipements, il était impératif d'automatiser le processus. L'avantage de l'utilisation d'un script est notamment que toutes les fiches d'identité peuvent être actualisées à un intervalle régulier et reflètent l'état des connaissances enregistrées dans la base de données. De plus, les nouveaux piézomètres sont automatiquement intégrés dans le processus, ce qui permet d'avoir des fiches à disposition pour l'ensemble des équipements.

A l'heure actuelle, deux principales limites peuvent être mentionnées : la pérennisation de l'automatisation au sein de l'Etat de Genève et le temps d'exécution. Le but était de transmettre le script aux entités de l'Etat de Genève (OCSIN et DIT), qui sont responsables pour l'exécution automatique de scripts en arrière-plan. Cependant, des questions liées à la responsabilité du suivi ainsi que d'ordre technique ont ralenti le processus, ce qui a empêché la réalisation avant la fin du stage. Une des problématiques était également, que le code avait été écrit sur Python et qu'il demandait l'activation d'un module supplémentaire (reportLab). A priori, il y aurait eu moins de problèmes si le travail avait été réalisé avec FME, qui est plus fréquemment utilisé au sein de l'administration cantonale. Pour l'instant, le script doit être exécuté depuis un poste individuel, soit manuellement soit à travers le planificateur des tâches.

L'autre limite concerne le temps d'exécution, qui est très important et dépasse les 16h pour créer toutes les 5000 fiches. La performance du script pourrait certainement être améliorée par un « data scientist », qui a des connaissances plus approfondies et pourrait optimiser le processus. Il reste à mentionner que la quasi-totalité des opérations a été réalisée avec des outils arcpy, qui lient ArcGIS Pro à Python, mais il est probable que d'autres modules auraient été plus performants dans le traitement des données (p.ex. pour les statistiques des mesures). En raison de cette limite liée au temps d'exécution, un script alternatif a été préparé qui permet de créer des fiches uniquement pour les nouveaux piézomètres. Dans l'idéal, toutes les fiches auraient dû être actualisées à un intervalle hebdomadaire, cependant ceci n'est pas envisageable avec le temps d'exécution actuel. Il a donc été décidé de faire tourner le script alternatif toutes les semaines et le script principal une fois par mois.

En consultant les fiches de piézomètres, il s'est avéré que de nombreux attributs n'étaient pas systématiquement renseignés. Ce manque d'information se répercute directement sur la qualité des fiches d'identités, qui sont nourries par les données de SOLSTISS. Cela souligne l'importance d'enrichir la base de données afin de mieux connaître les ressources et de pouvoir mieux gérer et exploiter les équipements à disposition. L'enregistrement systématique des projets de forages par les bureaux d'études sur internet, qui sera requis et opérationnel sous peu avec la mise en production du système SOLSTISS à plus large échelle, permettra de récupérer de nombreuses informations. A terme, le nombre de nouvelles données disponibles augmentera, mais il serait également souhaitable de rajouter et actualiser les informations pour les équipements déjà existants et exploités par le GESDEC. La possibilité de maintenir à jour les fiches à travers le script, est donc une des grandes forces du travail de stage proposé et permet aux fiches d'évoluer avec la base de données. Ceci garantit un rendu qui reflète l'état des connaissances les plus récentes, même au-delà de la fin du stage.

RÉFLEXIONS SUR LE DÉROULEMENT DU STAGE

Le stage de géomatique effectué au sein du GESDEC a été très bénéfique pour appliquer et développer mes compétences. L'intérêt et le soutien des collègues pour nos projets de stage a été très encourageant et a permis de développer un outil qui aura une utilité concrète pour le service à plus long terme. La mise à disposition d'une place de travail, avec tout le matériel informatique nécessaire, a offert un cadre de travail qui permettait de totalement se consacrer au projet. Tout au long de mon stage, j'ai pu bénéficier d'un très bon encadrement, pendant lequel j'ai toujours pu poser toutes mes questions et ai été soutenu dans mes démarches. J'ai particulièrement apprécié la grande autonomie dont j'ai pu profiter dans la réalisation de mon travail. En plus, la présence de ma costagiaire Lillia a favorisé l'entraide et l'échange sur des

sujets techniques. Une proposition d'amélioration de ma part serait l'instauration d'un point hebdomadaire, qui permettrait d'échanger sur l'avancée du projet et de distinguer de potentiels problèmes plus en amont dans le processus. En conclusion, je garderai un excellent souvenir de cette expérience enrichissante au sein du GESDEC et la conseille vivement à toute personne intéressée par la géomatique et la géologie.

5. CONCLUSION

Ce travail écrit a permis de présenter le processus de travail et le rendu de ce projet de stage en géomatique, qui a abouti à la création d'un script permettant la génération automatique de fiches d'identité pour les piézomètres. Les fiches d'identité synthétisent les informations attributaires provenant de la base de données de SOLSTISS et du serveur du GESDEC en forme de résumé en format PDF, qui convient aux besoins du métier. L'exécution du script permet d'actualiser l'ensemble des fiches et de maintenir les informations à jour. Ainsi, les fiches, qui sont librement accessibles sur le serveur du GESDEC, représentent un matériel de travail qui reflète les connaissances actuelles et facilite la consultation des informations pour chaque piézomètre. A travers la création du script et la génération des fiches de piézomètres, les objectifs formulés initialement ont été atteints. Malgré la persistance de certaines limites liées à la gestion et performance du script, les fiches d'identité sont à disposition du service et prêtes à l'emploi. En conclusion, ce travail met en lumière la plus-value qui découle d'un système d'information comme SOLSTISS, qui organise et met à disposition des géodonnées standardisées, qui peuvent être par la suite revalorisées dans un format transformé. Ce projet de stage démontre l'important potentiel des outils géomatiques à créer des produits personnalisés et automatisés qui permettent d'optimiser la mise à disposition de données en fonction de besoins précis.

BIBLIOGRAPHIE

Eaufrance (2019). Le niveau des nappes souterraines. https://www.eaufrance.fr/le-niveau-des-nappes-souterraines (consulté le 17.01.2022).

esri (s.d.). ArcMap: présentation des rapports dans ArcGIS. https://desktop.arcgis.com/fr/arcmap/10.4/map/reports/what-are-reports-in-arcgis.htm (consulté le 17.12.2021)

OFEV (2019). Réservoirs aquifères. https://www.bafu.admin.ch/bafu/fr/home/themes/eaux/info-specialistes/etat-des-eaux/etat-des-eaux-souterraines/nappes-d_eaux-souterraines.html (consulté le 16.01.2022).

OFEV (2020). Etat des eaux souterraines. https://www.bafu.admin.ch/bafu/fr/home/themes/eaux/info-specialistes/etat-des-eaux/etat-des-eaux-souterraines.html (consulté le 16.01.2022).

République et Canton de Genève. (s.d.-a). Les eaux souterraines. https://www.ge.ch/dossier/eau-potable-geneve/ressources-naturelles-notre-canton/eaux-souterraines (consulté le 16.01.2022).

République et Canton de Genève (s.d.-b). Le système d'information SOLSTISS. https://www.ge.ch/dossier/gestion-durable-ressources-du-sol/gerer-proteger-ressources-du-sol/systeme-information-solstiss (consulté le 18.01.2022).

République et Canton de Genève (2019). Office Cantonal de l'Environnement (Plaquette).

République et Canton de Genève (2020a). Plan de Gestion des ressources du sous-sol. https://www.ge.ch/document/24912/telecharger (consulté le 17.02.2022).

République et Canton de Genève (2020b). SOLSTISS, catalogue de base de données V05. GESDEC.

Syndicat Mixte pour la protection et la gestion des nappes souterraines de la plaine du Roussillon (s.d.). Nappes quaternaires. https://www.nappes-roussillon.fr/Nappes-quaternaires.html (consulté le 17.02.2022).

ANNEXES

- Annexe 1: Scripts Python
- Annexe 2: Fiches d'identité de piézomètres
- Annexe 3 : Manuel d'utilisation destiné au service (GESDEC)

```
#-*- coding: utf-8 -*-
# ---------------------------------------------------------------------------
## ResizeImage.py
# Description: Script pour l'adaptation de la taille des photos pour correspondre aux exigences du script CreationFiche_EquipementSOLSTISSMesures.py.
# Il utilise comme input les images enregistrées dans le fichier Original puis les transforme et les enregistre dans le fichier Resized.
# ---------------------------------------------------------------------------
### SCRIPT FOR ADAPTING THE SIZE OF THE PICTURES FROM THE PIEZOMETRES
print("-------------------------------------------------------------------------------")
print("SCRIPT FOR ADAPTING THE SIZE OF THE PICTURES FROM THE PIEZOMETRES")
print("-------------------------------------------------------------------------------")

### 1. IMPORT MODULES
print ("Import all required modules")
import arcpy
from PIL import Image
#Overwrite Output: permission to overwrite existing data
arcpy.env.overwriteOutput = True

# ---------------------------------------------------------------------------
### 2. DATA MANAGEMENT

# Input Data
inputPath = "S:\\UO5196\\80_SYS_INFORMATION\\SI_GEOLOGIE\\05_DGSI\\01_Applications_Metiers\\SI_Geologie\\Stages\\Fiche_identite_piezo\\01_FichesPIEZO\\02_RepertoirePhotos\\Original"
outputPath="S:\\UO5196\\80_SYS_INFORMATION\\SI_GEOLOGIE\\05_DGSI\\01_Applications_Metiers\\SI_Geologie\\Stages\\Fiche_identite_piezo\\01_FichesPIEZO\\02_RepertoirePhotos\\Resized"

# ---------------------------------------------------------------------------
### 3. TRANSFORMATION

# Define the workspace
arcpy.env.workspace = inputPath
# List the files in the workspace
photoFiles= arcpy.ListFiles()

# Define the standard height the photos should have after the transformation
baseheight= 1500

# Loop through all the photos found in the input folder (inputPath)
for photo in photoFiles:
    print("Processing " + str(photo))
    # Define the exact path to the picture being processed
    photoPath = arcpy.env.workspace + "\\" + str(photo)
    # Opening the picture in order to modify the size
    img= Image.open(photoPath)
    # Determine the ratio of the desired height towards the original measures of the picture
    hpercent= (baseheight/float(img.size[1]))
    # This ratio allows to compute the new width of the picture, which allows to maintain the aspect ratio
    wsize= int((float(img.size[0])*float(hpercent)))
    # Resizing of the pricture according to the desired size of the picture
    img=img.resize((wsize, baseheight), Image.ANTIALIAS)
    # Determine the location in which the new picture should be saved
    output = outputPath + "\\" +str(photo)
    # Save the image
    img.save(output)

print("The size of all photos has been successfully adjusted").
```

```
#-*- coding: utf-8 -*-
# ---------------------------------------------------------------------------
## CreationFichePiezo_Serveur.py
# Description: Script pour la création d'une fiche d'identité pour les piézomètres. Elle réunit une carte, un schema ainsi que toutes les caractéristiques techniques connues sur le piézomètre.
# La principale source de donnée est la base de données SOLSTISS et certains autres documents proviennent du serveur du GESDEC. Tous les liens sont axés sur le V: de SOLSTISS ainsi que sur le serveur.
# Les fiches sont disponibles en format PDF sur le serveur du GESDEC après exécution du script (Emplacement: cf. General Datapaths OUTPUT ci-dessous).
# ---------------------------------------------------------------------------
### SCRIPT FOR THE CREATION OF PIEZOMETRE IDENTITY FILES
print("-------------------------------------------------------------------------------")
print("SCRIPT FOR THE CREATION OF PIEZOMETRE IDENTITY FILES")
print("-------------------------------------------------------------------------------")

### 1. IMPORT MODULES
print ("Import all required modules")
# Import arcpy module
import arcpy
import os
import reportlab
from arcpy.sa import *
#Overwrite Output: permission d'écraser des données antécédentes
arcpy.env.overwriteOutput = True

# Import reportlab
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import A4, portrait
from reportlab.lib import colors
from reportlab import platypus
from reportlab.lib.styles import ParagraphStyle as PS
from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph

#Import PIL
from PIL import Image

#Import date
from datetime import date

# ---------------------------------------------------------------------------

### 2. DATA MANAGEMENT

#FOR THE COMPILATION TABLE
print("-------------------------------------------------------------------------------")
print ("Set the data input and output paths")
##Set the transferGDBAttributeProperties environment to True
arcpy.env.transferDomains = True

### General DataPaths INPUT
#Geodatabase SOLSTISS
dataPath= "V:\Donnees_Applicatives\GEOMATIQUE\INTRANET\GEOTHERMIE2020_SOLSTISS\PROD\GEOTHERMIE2020_SOLSTISS.gdb\\"
#Links to ArcGIS projects
ProjetTEST = "S:\\UO5196\\80_SYS_INFORMATION\\SI_GEOLOGIE\\05_DGSI\\01_Applications_Metiers\\SI_Geologie\\Stages\\Fiche_identite_piezo\\01_FichesPIEZO\\06_ProjetsArcGIS\\ProjetTEST\\"
ProjetTEST_GDB =ProjetTEST + "ProjetTEST.gdb"
Piezometres_Map = "S:\\UO5196\\80_SYS_INFORMATION\\SI_GEOLOGIE\\05_DGSI\\01_Applications_Metiers\\SI_Geologie\\Stages\\Fiche_identite_piezo\\01_FichesPIEZO\\06_ProjetsArcGIS\\Piezometres_Map\\"

### General Datapaths OUTPUT
# Define the path to the ressources saved on the data server (Output)
dataserver = "S:\\UO5196\\80_SYS_INFORMATION\\SI_GEOLOGIE\\05_DGSI\\01_Applications_Metiers\\SI_Geologie\\Stages\\Fiche_identite_piezo\\01_FichesPIEZO\\"
outputpath = dataserver + "05_OutputFiches\\"
```

```
59
60  ### Input Data
61  #From SOLSTISS:
62  GEOL_SS_SONDAGE = dataPath + "GEOLOGIE_SOUS_SOL\\GEOL_SS_SONDAGE"
63  GEOL_SS_EQUIPEMENT = dataPath + "GEOL_SS_EQUIPEMENT"
64  GEOL_SS_ADMIN = dataPath + "GEOL_SS_ADMIN"
65  #GEOL_SS_ADRESSE = dataPath + "GEOL_SS_ADRESSE"
66  GEOL_SS_RESSOURCE = dataPath + "GEOL_SS_RESSOURCE"
67  #GEOL_SS_SONDAGE_MESURE: Problématique tant que la colonne "ID_EQUIPEMENT n'y figure
    pas dans le V:\. Dès qu'elle aura été activé, c'est la ligne de code ci-dessous qui
    devra être activé et celle du C:\ effacée.)
68  #GEOL_SS_SONDAGE_MESURE = dataPath + "GEOL_SS_SONDAGE_MESURE"
69  GEOL_SS_SONDAGE_MESURE =
    "C:\\Users\MARTIGI\\AppData\\Roaming\\Esri\\ArcGISPro\\Favorites\\SOLSTISS.sde\\GEO202
    0.GEOL_SS_SONDAGE_MESURE"
70  #Manually created:
71  SONDE_AUTO = ProjetTEST + "SONDE_AUTOMATIQUE.xlsx\\Piezo_SondeAutomatique$"
72  ADRESSES_PIEZO = ProjetTEST + "Adresses_Piezo.xls\\Adresses_Piezo$"
73
74  #From the server:
75  # Define path to the schemata ("coupe")
76  schema1 = dataserver + "03_Schema\\AltEndessousTerrainMOD.png"
77  schema2 = dataserver + "03_Schema\\AltAudessusTerrainMOD.png"
78  # Define the path to the pictures
79  photos = dataserver + "02_RepertoirePhotos\\Resized"
80  maps = dataserver + "01_RepertoireMaps\\"
81  logo = dataserver + "02_RepertoirePhotos\\Logo\\GeneveNoirTransp.png"
82
83  #From the internet/server:
84  # Define paths to the geological surveys
85  sondagesprive =
    "S:\\UO5196\\80_SYS_INFORMATION\\SI_GEOLOGIE\\FICHES_SONDAGES\\PRIVE\\"
86  sondagespublic = "https://ge.ch//sitg//geodata//SITG//SONDAGES_GEOLOGIQUES//"
87
88  #WorkingLayers (Assignation of names and paths for intermediate outputs)
89  Sondages = ProjetTEST_GDB  + "\\" + "Sondages"
90  Piezo = ProjetTEST_GDB  + "\\" + "Piezo" # Path to the compiled table in the
    GEODATABASE (Assembled from attribute tables, input data for the PDF File)
91  outputname = ProjetTEST + "Piezo_Copy.xlsx"
92  Sondages_Lyr = ProjetTEST_GDB + "\\" + "Sondages_Lyr"
93  Mesures_Copy = ProjetTEST_GDB + "\\" + "Mesures_Copy"
94  MESURE_Stat = ProjetTEST_GDB + "\\" + "MESURE_Stat"
95  Last_Measure_Date = ProjetTEST_GDB + "\\" + "Last_Measure_Date"
96  Date_Meas = ProjetTEST_GDB + "\\" + "Date_Meas"
97  GEOL_SS_EQUIPEMENT_table = ProjetTEST + "\\" +"GEOL_SS_EQUIPEMENT_table.xlsx"
98  GEOL_SS_EQUIPEMENT_tableF = ProjetTEST + "\\" + "GEOL_SS_EQUIPEMENT_table.xlsx" +
    "\\" + "GEOL_SS_EQUIPEMENT_table$"
99
100
101 # ------------------------------------------------------------------------------
102 ### 3. DATA PREPARATION: CREATION OF THE WORKING INPUT
103
104 ##INFORMATION FROM THE DATASERVERS AND OTHERS: COMPILATION
105 print("-------------------------------------------------------------------------")
106 print("Compilation of the relevant data for the input table")
107 #Define working environment
108 arcpy.env.workspace = ProjetTEST
109
110 #Recompilation of an Attribute Table for Piezometres
111
112 #Copy Piezo Shapefile to Working GDB
113 print ("- Copy Shapefile to Working GDB")
114 arcpy.management.CopyFeatures(GEOL_SS_SONDAGE, Sondages, '', None, None, None)
115
116 #Delete Irrelevant Fields for our purpose in the table serving as basic entity
117 print ("- Delete chosen Attributes")
118 arcpy.management.DeleteField(Sondages,
    "OBJECTIF;METHODE;TECHNIQUE;ALT_FOND_SONDAGE;PROFONDEUR_TVD;UNITE_GEOLOGIQUE_ATTEINTE;
    ORIENTATION;DATE_SAISIE;AUTEUR_SAISIE;DATE_MAJ;AUTEUR_MAJ;REMARQUE;DELTA_MODELE_TERRAI
    N;VALEUR_MODELE_TERRAIN;GEOLOGUE_RESPONSABLE")
119
120 #Rajout de l'attribut qui permet de faire la sélection dans sondage:
```

```
print ("- Add Information from GEOL_SS_EQUIPEMENT")
arcpy.conversion.TableToExcel(GEOL_SS_EQUIPEMENT, GEOL_SS_EQUIPEMENT_table, "NAME", "DESCRIPTION")
arcpy.management.JoinField(Sondages, "ID_ADMIN", GEOL_SS_EQUIPEMENT_tableF, "ID_ADMIN", ["TYPE_EQUIPEMENT", "ID_TEMP_SP", "TOURNEE", "DATE_NIVELLEMENT", "DISTANCE_A_PIEDS", "CONDITION_ACCES", "OUVERTURE", "USAGE", "PROFONDEUR_EQUIPEMENT", "ALT_REFERENTIEL",

   "TUBAGE", "DIAMETRE_TUBE", "HAUTEUR_CREPINE", "BASE_CREPINE", "SONDE_ENREGISTREUSE", "REFERENTIEL", "ID_EQUIPEMENT", "ETAT_EQUIPEMENT","NOM_PIEZO_TEMP"])
arcpy.management.MakeFeatureLayer(in_features=Sondages, out_layer=Sondages_Lyr)

#Créer une sélection pour uniquement garder les piézomètres
arcpy.management.SelectLayerByAttribute(Sondages_Lyr, "NEW_SELECTION", "TYPE_EQUIPEMENT = 'Piézomètre'", None)
arcpy.management.CopyFeatures(Sondages_Lyr, Piezo)

#Add Information from other Tables keeping the domains:
print ("- Add Information from GEOL_SS_ADMIN")
arcpy.management.JoinField(Piezo, "ID_ADMIN" , GEOL_SS_ADMIN, "ID_ADMIN", ["ID_PROJET", "FIN_TRAVAUX"])

#GEOL_SS_RESSOURCE
print ("- Add Information from GEOL_SS_RESSOURCE")
#Add Information for Fields where only number has been added (due to the domain, circumvent by Table to Table):
arcpy.conversion.TableToTable(GEOL_SS_RESSOURCE, ProjetTEST_GDB, "CopyRessource")
CopyRessource = ProjetTEST_GDB + "\\" + "CopyRessource"
arcpy.management.JoinField(Piezo, "ID_ADMIN", CopyRessource, "ID_ADMIN", ["NOM", "NAPPE", "PRESSION_NAPPE", "UNITE_GEOL_PRINCIPALE", "UNITE_GEOL_SECONDAIRE"])


#ADRESSES_PIEZO
print ("- Add Adresses from ADRESSES_PIEZO")
arcpy.management.JoinField(Piezo, "ID_EQUIPEMENT", ADRESSES_PIEZO, "ID_EQUIPEMENT", ["Adresse_Piezo", "NoPostal_Piezo", "Commune_Piezo"])

# ATTENTION: sera à terme evtl. intégré aux adresses de projet ci-après (GEOL_SS_ADRESSE) et il faudra donc activer la ligne ci-dessous et retirer celle en-haut (#ADRESSES_PIEZO)
#GEOL_SS_ADRESSE
#print ("- Add Information from GEOL_SS_ADRESSE")
#arcpy.management.JoinField(Piezo, "ID_PROJET" , GEOL_SS_ADRESSE, "ID_PROJET", ["ADRESSE_RUE", "ADRESSE_NUM_RUE", "ADRESSE_NPA", "ADRESSE_VILLE"])

#SONDE_AUTO
print ("- Add Information from SONDE_AUTO")
arcpy.management.JoinField(Piezo, "ID_EQUIPEMENT", SONDE_AUTO, "ID_EQUIPEMENT", ["AUTOMATIQUE", "LIEN_VH"])

#Delete intermediate outputs
print ("- Delete intermediate output")
arcpy.management.Delete(CopyRessource)
arcpy.management.Delete(GEOL_SS_EQUIPEMENT_table)
print ("Table has been compiled successfully")

##DETERMINATION OF THE LASTEST MEASURES AND RECORDS
print("--------------------------------------------------------------------------------")
print("Determination of the latest measures and records in the database")
#Define working environment
arcpy.env.overwriteOutput = True
arcpy.env.workspace = ProjetTEST_GDB

#Creation of the Summary
arcpy.analysis.Statistics(GEOL_SS_SONDAGE_MESURE, MESURE_Stat, "NIVEAU_NAPPE MEAN;NIVEAU_NAPPE MIN;NIVEAU_NAPPE MAX;TEMPERATURE MEAN; NIVEAU_MESURE COUNT;TEMPERATURE COUNT", "ID_EQUIPEMENT")
arcpy.analysis.Statistics(GEOL_SS_SONDAGE_MESURE, Last_Measure_Date, "DATE_MESURE MAX", "ID_EQUIPEMENT;TYPE_MESURE")
arcpy.analysis.TableSelect(Last_Measure_Date, Date_Meas, "TYPE_MESURE = 2")
arcpy.management.JoinField(MESURE_Stat, "ID_EQUIPEMENT", Date_Meas, "ID_EQUIPEMENT",
```

```
    "MAX_DATE_MESURE")
174
175 #Creation of new columns to receive the data
176 arcpy.management.AddField(MESURE_Stat, "MIN_Date", "DATE")
177 arcpy.management.AddField(MESURE_Stat, "MAX_Date", "DATE")
178 arcpy.management.AddField(MESURE_Stat, "DER_NIVEAU_NAPPE", "FLOAT")
179
180
181 print("Data selection and definition")
182 fieldsStat = ["ID_EQUIPEMENT", "MIN_NIVEAU_NAPPE", "MAX_NIVEAU_NAPPE",
    "MAX_DATE_MESURE", "MIN_Date", "MAX_Date", "DER_NIVEAU_NAPPE"]
183 fieldsMes =["ID_EQUIPEMENT", "DATE_MESURE", "NIVEAU_NAPPE", "TYPE_MESURE"]
184 def unique_values(table, field):
185     with arcpy.da.SearchCursor(MESURE_Stat, "ID_EQUIPEMENT") as cursor:
186         return sorted({row[0] for row in cursor})
187 piezos_list = unique_values(MESURE_Stat, "ID_EQUIPEMENT")
188
189 for i in piezos_list:
190     expression = '"ID_EQUIPEMENT" = ' + str(i)
191     # Grasp the data for each piezometre
192     #First cursor
193     with arcpy.da.SearchCursor(MESURE_Stat, fieldsStat, expression) as cursor:
194         for row in cursor:
195             ID_EQUIPEMENT = str(row[0])
196             MIN_NAPPE = row[1]
197             MAX_NAPPE = row[2]
198             DERMESURE_Date = row[3] #Dernière mesure nappe
199
200         #Search in data corresponding value
201         if MIN_NAPPE is not None:
202             MIN_NAPPE = str("{:.2f}".format(float(MIN_NAPPE)))
203             expression1 = '"ID_EQUIPEMENT" = '+ str(i) + " AND " + '"NIVEAU_NAPPE" =
                ' + MIN_NAPPE
204             list1=[] # Create a list with all the dates when the minimum date is
                being met
205             with arcpy.da.SearchCursor(GEOL_SS_SONDAGE_MESURE, fieldsMes,
                expression1) as cursor:
206                 for row in cursor:
207                     MIN_Date = row[1]
208                     list1.append(MIN_Date)
209             if list1: # should be met when the list is not emtpy
210                 latest_min= max(list1)
211             else:
212                 latest_min = None
213
214         if MAX_NAPPE is not None:
215             MAX_NAPPE = str("{:.2f}".format(float(MAX_NAPPE)))
216             expression2 = '"ID_EQUIPEMENT" = '+ str(i) + " AND " + '"NIVEAU_NAPPE" =
                ' + MAX_NAPPE
217             list2=[] #necessity to create a list as sometimes the maximum is being
                reached at different dates, the lastest one is being chosen here
218             with arcpy.da.SearchCursor(GEOL_SS_SONDAGE_MESURE, fieldsMes,
                expression2) as cursor:
219                 for row in cursor:
220                     MAX_Date = row[1]
221                     list2.append(MAX_Date)
222             if list2: # should be met when the list is not emtpy
223                 latest_max= max(list2)
224             else:
225                 latest_max = None
226
227         if DERMESURE_Date is not None:
228             DERMESURE_Date = str(DERMESURE_Date)
229             expression3 = '"ID_EQUIPEMENT" = '+ str(i) + ' AND "DATE_MESURE"=
                timestamp' + '\''+ str(DERMESURE_Date) + '\''+ " AND " + '"TYPE_MESURE"
                ='+ str(2) #Correspond au type de mesure Relevé eau
230             with arcpy.da.SearchCursor(GEOL_SS_SONDAGE_MESURE, fieldsMes,
                expression3) as cursor:
231                 for row in cursor:
232                     DER_NIVEAU_NAPPE = row[2]
233                     if DER_NIVEAU_NAPPE is not None:
234                         DER_NIVEAU_NAPPE =
```

```
                            str("{:.2f}".format(float(DER_NIVEAU_NAPPE)))
        if DERMESURE_Date is None:
            DER_NIVEAU_NAPPE = None


        with arcpy.da.UpdateCursor(MESURE_Stat, fieldsStat, expression) as cursor:
            for row in cursor:
                row[4]=latest_min
                row[5]=latest_max
                row[6]=DER_NIVEAU_NAPPE
                cursor.updateRow(row)

#Join information from measures to the Piezo table
print ("- Add Information from MESURE_Stat")
arcpy.management.JoinField(Piezo, "ID_EQUIPEMENT", MESURE_Stat, "ID_EQUIPEMENT",
["MEAN_NIVEAU_NAPPE", "MIN_NIVEAU_NAPPE", "MIN_Date", "MAX_NIVEAU_NAPPE",
"MAX_Date", "DER_NIVEAU_NAPPE", "MAX_DATE_MESURE", "MEAN_TEMPERATURE",
"COUNT_NIVEAU_MESURE", "COUNT_TEMPERATURE"])


# TRANSFORMATION INTO AN EXCEL TO FACILITATE LATER WORK
# COPY THE GEODATA INFORMATION INTO EXCEL
# (otherwise problems with data linked to domains, providing the coded number
instead of the descriptions)
print("Copy of the table into Excel format")
arcpy.conversion.TableToExcel(Piezo, outputname, "NAME", "DESCRIPTION")

# ---------------------------------------------------------------------------

## PREPARATION OF THE DATA FOR THE MAP
# Process: Make Feature Layer (Make Feature Layer)
print("------------------------------------------------------------------------------")
print ("Data Preparation for the map")
Output_Layer = "Piezo"
arcpy.MakeFeatureLayer_management(in_features=Piezo, out_layer=Output_Layer,
where_clause="", workspace="", field_info="OBJECTID OBJECTID VISIBLE NONE;SHAPE
SHAPE VISIBLE NONE;ID_EQUIPEMENT ID_EQUIPEMENT VISIBLE NONE;ID_SONDAGE ID_SONDAGE
VISIBLE NONE;NOM_PIEZO_TEMP NOM_PIEZO_TEMP VISIBLE NONE;TOURNEE TOURNEE VISIBLE
NONE;NO_PRIVE NO_PRIVE VISIBLE NONE;ALT_TERRAIN ALT_TERRAIN VISIBLE
NONE;PROFONDEUR_MD PROFONDEUR_MD VISIBLE NONE;ID_ADMIN ID_ADMIN VISIBLE NONE;COORD_X
COORD_X VISIBLE NONE;COORD_Y COORD_Y VISIBLE NONE;UNITE_GEOLOGIQUE_ATTEINTE
UNITE_GEOLOGIQUE_ATTEINTE VISIBLE NONE;DISTANCE_A_PIEDS DISTANCE_A_PIEDS VISIBLE
NONE;CONDITION_ACCES CONDITION_ACCES VISIBLE NONE;OUVERTURE OUVERTURE VISIBLE
NONE;USAGE USAGE VISIBLE NONE;PROFONDEUR_EQUIPEMENT PROFONDEUR_EQUIPEMENT VISIBLE
NONE;ALT_REFERENTIEL ALT_REFERENTIEL VISIBLE NONE;TUBAGE TUBAGE VISIBLE
NONE;DIAMETRE_TUBE DIAMETRE_TUBE VISIBLE NONE;HAUTEUR_CREPINE HAUTEUR_CREPINE
VISIBLE NONE;BASE_CREPINE BASE_CREPINE VISIBLE NONE;SONDE_ENREGISTREUSE
SONDE_ENREGISTREUSE VISIBLE NONE;REMARQUE REMARQUE VISIBLE NONE;DATE_NIVELLEMENT
DATE_NIVELLEMENT VISIBLE NONE;REFERENTIEL REFERENTIEL VISIBLE NONE;ID_EQUIPEMENT
ID_EQUIPEMENT VISIBLE NONE;ETAT_EQUIPEMENT ETAT_EQUIPEMENT VISIBLE NONE;ID_TEMP_SP
ID_TEMP_SP VISIBLE NONE;ID_PROJET ID_PROJET VISIBLE NONE;ADRESSE_RUE ADRESSE_RUE
VISIBLE NONE;ADRESSE_NPA ADRESSE_NPA VISIBLE NONE;ADRESSE_VILLE ADRESSE_VILLE
VISIBLE NONE;ADRESSE_NUM_RUE ADRESSE_NUM_RUE VISIBLE NONE;TYPE_CONTACT TYPE_CONTACT
VISIBLE NONE;ID_CONTACT ID_CONTACT VISIBLE NONE;RAISON_SOCIALE RAISON_SOCIALE
VISIBLE NONE;EMAIL EMAIL VISIBLE NONE;TELEPHONE TELEPHONE VISIBLE NONE;ADRESSE_VILLE
ADRESSE_VILLE VISIBLE NONE;NAPPE NAPPE VISIBLE NONE;PRESSION_NAPPE PRESSION_NAPPE
VISIBLE NONE;UNITE_GEOL_PRINCIPALE UNITE_GEOL_PRINCIPALE VISIBLE
NONE;UNITE_GEOL_SECONDAIRE UNITE_GEOL_SECONDAIRE VISIBLE NONE;REMARQUE_1 REMARQUE_1
VISIBLE NONE;Automatique Automatique VISIBLE NONE;Lien_VH Lien_VH VISIBLE
NONE;Adresse_Piezo Adresse_Piezo VISIBLE NONE;NoPostal_Piezo NoPostal_Piezo VISIBLE
NONE;Commune_Piezo Commune_Piezo VISIBLE NONE")

# Process: Save To Layer File (Save To Layer File)
Piezo_Layer = Piezometres_Map + "Piezo.lyrx"
arcpy.SaveToLayerFile_management(in_layer=Output_Layer, out_layer=Piezo_Layer,
is_relative_path="", version="CURRENT")

# Create Basic Map
print("------------------------------------------------------------------------------")
print("Creation of the basic map")
# Provide link to existing project
aprx= arcpy.mp.ArcGISProject(Piezometres_Map + "Piezometres_Map.aprx")
```

```
275
276 # List the maps in the project
277 m=aprx.listMaps("Map")[0]
278 m2=aprx.listMaps("Geneve")[0]
279
280
281 # Define the layer file and add it to the map
282 lyrFile= arcpy.mp.LayerFile(Piezo_Layer)
283 m.addLayer(lyrFile,"TOP")
284 lyr=m.listLayers()[0]
285 if lyr.name == "Piezo":
286     lyr.name = "Piézomètres"
287
288 # Define Symbology
289 sym = lyr.symbology
290 sym.updateRenderer('SimpleRenderer')
291 if sym.renderer.type == 'SimpleRenderer':
292     sym.renderer.symbol.size = 15
293     sym.renderer.symbol.color = {'RGB' : [30, 150, 203, 100]}
294     lyr.symbology = sym
295
296 aprx.save()
297
298 # Define function to create Map
299 print("--------------------------------------------------------------------------------")
300 print ("Definition of the specific map function to locate individual piezometres")
301
302 def create_map(ID_EQUIPEMENT):
303     # CREATE MAP
304
305     # Process: Select Layer By Attribute (Select Layer By Attribute)
306     criteria = "ID_EQUIPEMENT = " +  str(ID_EQUIPEMENT)
307     arcpy.SelectLayerByAttribute_management(in_layer_or_view=lyr,
        selection_type="NEW_SELECTION", where_clause=criteria, invert_where_clause="")
308
309     # Export Selection to a new Layer
310     Piezo_Selection1 = Piezometres_Map + "PiezoSelection1.lyrx"
311     arcpy.MakeFeatureLayer_management(in_features=lyr, out_layer=Piezo_Selection1,
        where_clause=criteria, workspace="",field_info="OBJECTID OBJECTID VISIBLE
        NONE;SHAPE SHAPE VISIBLE NONE;ID_EQUIPEMENT ID_EQUIPEMENT VISIBLE
        NONE;ID_SONDAGE ID_SONDAGE VISIBLE NONE")
312     Piezo_Selection2 = Piezometres_Map + "PiezoSelection2.lyrx"
313     arcpy.SaveToLayerFile_management(in_layer=Piezo_Selection1,
        out_layer=Piezo_Selection2, is_relative_path="", version="CURRENT")
314
315     selectionFile= arcpy.mp.LayerFile(Piezo_Selection2)
316     m.addLayer(selectionFile,"TOP")
317     lyr2=m.listLayers("*PiezoSelection*")[0]
318
319     # Define Symbology for the selected piezometre
320     sym2 = lyr2.symbology
321     sym2.updateRenderer('SimpleRenderer')
322     if sym2.renderer.type == 'SimpleRenderer':
323         symbolList2 = sym2.renderer.symbol.listSymbolsFromGallery('')
324         symbol2 = symbolList2[55]
325         sym2.renderer.symbol = symbol2
326         sym2.renderer.symbol.size = 50
327         sym2.renderer.symbol.color = {'RGB' : [200, 0, 10, 100]}
328         lyr2.symbology = sym2
329
330     #Define Symbology for the Overview Map (Geneva)
331     m2.addLayer(selectionFile,"TOP")
332     lyr3=m2.listLayers("*PiezoSelection*")[0]
333
334     # Define Symbology for the position of the selected piezometre in the overview map
335     sym3 = lyr3.symbology
336     sym3.updateRenderer('SimpleRenderer')
337     if sym3.renderer.type == 'SimpleRenderer':
338         symbolList3 = sym3.renderer.symbol.listSymbolsFromGallery('')
339         symbol3 = symbolList3[55]
340         sym3.renderer.symbol = symbol3
341         sym3.renderer.symbol.size = 30
```

```
            sym3.renderer.symbol.color = {'RGB' : [200, 0, 10, 70]}
            lyr3.symbology = sym3

        # List the layouts in the project
        lyt= aprx.listLayouts("Layout2")[0]

        # Define the mapframes in the layout
        mf= lyt.listElements("MAPFRAME_ELEMENT", "Map*")[0]
        mf.camera.setExtent(mf.getLayerExtent(lyr, True, True))
        mf.camera.scale=1500

        #Export Map
        outputname = maps + "map" + str(ID_EQUIPEMENT) + ".png"
        lyt.exportToPNG(outputname, 150)

        # Clear Selection
        arcpy.SelectLayerByAttribute_management(in_layer_or_view=lyr,
        selection_type="CLEAR_SELECTION")

        # Remove Layer (so a new one will be added for the next utilisation)

        m.removeLayer(lyr2)
        m2.removeLayer(lyr3)
        arcpy.Delete_management(Piezo_Selection2)
        # Save the project
        aprx.save()

# ---------------------------------------------------------------------------

### 4. DATA SELECTION AND DEFINITION FOR THE PDF REPORT

### DEFINITION OF THE DATA REQUIRED
# Defines the fields that are the inputs to SearchCursor
print("-------------------------------------------------------------------------")
print("Data selection and definition for the PDF report")
fc = outputname + "\\Piezo_Copy$"
fields = ["ID_EQUIPEMENT", "ID_SONDAGE", "NOM_PIEZO_TEMP", "ID_TEMP_SP", "COORD_X",
"COORD_Y", "Adresse_Piezo", "NoPostal_Piezo", "Commune_Piezo", "DISTANCE_A_PIEDS",
          "CONDITION_ACCES", "OUVERTURE", "ETAT_EQUIPEMENT", "USAGE", "TOURNEE",
          "FIN_TRAVAUX", "DIAMETRE_FOND_TROU", "ALT_TERRAIN",
          "ALT_REFERENTIEL", "REFERENTIEL", "DATE_NIVELLEMENT",
          "PROFONDEUR_EQUIPEMENT", "PROFONDEUR_MD", "TUBAGE", "DIAMETRE_TUBE",
          "HAUTEUR_CREPINE", "BASE_CREPINE",
          "SONDE_ENREGISTREUSE", "NOM", "NAPPE", "PRESSION_NAPPE",
          "UNITE_GEOL_PRINCIPALE", "UNITE_GEOL_SECONDAIRE", "Automatique",
          "Lien_VH", "COUNT_NIVEAU_MESURE",
          "MEAN_NIVEAU_NAPPE", "MIN_NIVEAU_NAPPE", "MAX_NIVEAU_NAPPE",
          "MEAN_TEMPERATURE", "MAX_DATE_MESURE", "MIN_Date", "MAX_Date",
          "DER_NIVEAU_NAPPE", "COUNT_TEMPERATURE"]

# Preparation of the loop: List all  Piezometres in Geneva for the loop
def unique_values(table , field):
    with arcpy.da.SearchCursor(fc, [field]) as cursor:
        return sorted({row[0] for row in cursor})
piezos = unique_values(fc, "ID_EQUIPEMENT")

# Modification of the format (from float to integer)
for i in range(len(piezos)):
    piezos[i]=int(piezos[i])

### FUNCTIONS COMPOSING THE CANVAS

#Define positions
print("-------------------------------------------------------------------------")
print ("Definition of orientation points on the canvas")

def define_positions():
    global x1
    global x11
    global x2
    global x22
    global xIT
```

```
405         global xIR
406         global xM1
407         global xM2
408         global y
409         global yS1
410         global yR
411         global yS2
412         global widthH
413         global widthF
414         global dist
415         global yM
416         global xP1
417         global xP2L
418         global xP2P
419         global yP
420         global yPL
421         global yPP
422         global yCT
423         global yIT
424         global yIR
425
426
427         # Define distances
428         # x-Positioning (horizontal position)
429         x1= 315      #Start of text on the right of the map
430         x11=440      #Start of text containing the information right on the map
431         x2= 30       #Start of text of the information on the left side of the page
            (Technical characteristics)
432         x22= 170     #Start of the text containing the information on the left
433         xM1 = x1 + 20
434         xM2 = x11
435
436         # y-Positioning (vertical position)
437         y = 810       #Position of the titel
438         yS1 = 757    #Start of the localization text (Page 1)
439         yCT=465      #Start of the technical caracteristics
440         yR = 285     #Position resources when on page 1
441         yS2 = 755    #Start of the resources text (Page 2)
442
443         # Layout positions
444         widthH = 265 #Width Half of Illustrations
445         widthF = 545
446         dist = 25    #Distance from the left for illustrations
447
448         # Map position
449         yM = -316
450
451         #Schema
452         yIT = 227    #Version avec Terrain > Referentiel
453         xIT = 505
454         xIR = 505  #Version avec Referentiel > Terrain
455         yIR = 227
456
457         #Photos
458         xP1 = dist #Photo 1
459         xP2L = 305 #Photo 2 Landscape
460         xP2P = 355 #Photo 2 Portrait
461         yP = -580  #Position of the photos
462         yPL = 71   #If Landscape (for frame)
463         yPP = 26   #If Portrait (for frame)
464
465
466     #Define Function to create Title
467     print("------------------------------------------------------------------------")
468     print("Define functions:")
469     print ("- title")
470     def title_function(y):
471         #Set Box around title
472         canvas.setLineWidth(0.3)
473         canvas.setStrokeColorRGB(0.5, 0.5, 0.5, alpha =1)
474         canvas.setFillColorRGB(0.2, 0.65, 0.9, alpha = 0.2)
475         # Draw a rectangle
```

```
476         canvas.rect(15, 780, 565, 50, stroke = 1, fill = 1)
477         canvas.setFillColorRGB(1, 1, 1, alpha= 1)
478 
479         # Set Title
480         canvas.setFont('Helvetica-Bold', 14)
481         canvas.setFillColorRGB(0, 0, 0)
482         canvas.drawString(110,y, "Piézomètre ID_EQUIPEMENT " + ID_EQUIPEMENT)
483         canvas.drawString(x11,810, "Sondage " + ID_SONDAGE)
484         y-=20
485         canvas.setFont('Helvetica', 12)
486         if row[2] is not None:
487             canvas.drawString(110,y, NOM_PIEZO_TEMP)
488         if row[3] is not None:
489             canvas.drawString(x11, y, "OSite " + ID_TEMP_SP)
490         y-=30
491 
492         # Add Logo
493         canvas.drawImage(logo, 30, 634, width=58, preserveAspectRatio=True, mask = 'auto')
494 
495     print ("- localisation")
496     def localisation_function(NPA):
497         global y
498         canvas.setStrokeColorRGB(0, 0, 0)
499         canvas.setFillColorRGB(0,0,0)
500         canvas.setFont('Helvetica-Bold', 10)
501         y= yS1
502         canvas.drawString(x1, y, "Localisation")
503         canvas.setFont('Helvetica', 10)
504         y-=20
505         canvas.drawString(x1, y, "Coord. X [CH1903+]: ")
506         listX= COORD_X.split(".")
507         canvas.drawString(x11, y, listX[0])
508         y-=15
509         canvas.drawString(x1, y, "Coord. Y [CH1903+]: ")
510         listY=COORD_Y.split(".")
511         canvas.drawString(x11, y, listY[0])
512         y-=15
513         canvas.setFont('Helvetica', 10)
514         canvas.drawString(x1, y, "Adresse:")
515         if row[6] is None:
516             if row[7] is not None and row[8] is not None:
517                 NPA=str(int(float(row[7])))
518                 canvas.drawString(x11, y, str(NPA) + " " + str(COMMUNE))
519                 y-=25
520             else:
521                 canvas.drawString(x11, y, "-")
522                 y-=25
523         if row[6] is not None:
524             x= len(RUE)
525             if x <22:
526                 canvas.drawString(x11, y, RUE)
527             else:
528                 split = RUE.rsplit(' ',2)
529                 n= len(split)
530                 road = split[0]
531                 if n <3:
532                     number = split[1]
533                 if n ==3:
534                     number = split[1] + " " + split[2]
535                 if n >3:
536                     number = split[1] + " " + split[2] + " " + split[3]
537                 canvas.drawString(x11, y, str(road))
538                 y-=15
539                 canvas.drawString(x11, y, str(number))
540             if row[7] is not None and row[8] is not None:
541                 NPA=str(int(float(row[7])))
542                 y1= y-15
543                 canvas.drawString(x11, y1, str(NPA) + " " + str(COMMUNE))
544                 y-=40
545 
546     print ("- access")
547     def access_function():
```

```
548         global y
549         global yOuverture
550         global ybox1
551         canvas.setFont('Helvetica-Bold', 10)
552         canvas.drawString(x1, y, "Accès")
553         canvas.setFont('Helvetica', 10)
554         y-=20
555         canvas.drawString(x1, y,"Distance à pieds: ")
556         if row[9] is not None:
557             canvas.drawString(x11, y, DISTANCE_A_PIEDS)
558         else:
559             canvas.drawString(x11, y, "-")
560         y-=15
561         canvas.drawString(x1, y,"Condition d'accès: ")
562         if row[10] is not None:
563             canvas.drawString(x11, y, CONDITION_ACCES)
564         else:
565             canvas.drawString(x11, y,"-")
566         y-=15
567         canvas.drawString(x1, y,"Ouverture: ")
568         ybox1 = y-7
569         if row[11] is not None:
570             canvas.drawString(x11, y, OUVERTURE)
571         else:
572             canvas.drawString(x11, y,"-")
573 
574     print ("- activity")
575     def activity_function():
576         global y
577         canvas.setFont('Helvetica-Bold', 10)
578         y-=25
579         canvas.drawString(x1, y, "Activité ")
580         canvas.setFont('Helvetica', 10)
581         y-=20
582         canvas.drawString(x1, y,"Etat de l'ouvrage: ")
583         if row[12] is not None:
584             canvas.drawString(x11, y, ETAT_EQU)
585         else:
586             canvas.drawString(x11, y, "-")
587         y-=15
588         canvas.drawString(x1, y, "Réalisation du sondage: ") # correspond au critère fin
        travaux (GEOL_SS_SONDAGE)
589         if row[15] is not None:
590             day = FIN_TRAVAUX.day
591             if day <10:
592                 day = "0" + str(day)
593             month = FIN_TRAVAUX.month
594             if month < 10:
595                 month = "0" + str(month)
596             year = FIN_TRAVAUX.year
597             canvas.drawString(x11, y, str(day) + "." + str(month) + "." + str(year))
598         else:
599             canvas.drawString(x11, y, "-")
600         y-=15
601         canvas.drawString(x1, y,"Usage: ")
602         if row[13] is not None:
603             canvas.drawString(x11, y, USAGE)
604         else:
605             canvas.drawString(x11, y, "-")
606         y-=15
607         if row[14] is not None:
608             canvas.drawString(x1, y,"Tournée (GESDEC): ")
609             canvas.drawString(x11, y,TOURNEE)
610             y-=15
611             canvas.setFont('Helvetica', 10)
612             canvas.drawString(x1, y,"Contact: ")
613             canvas.drawString(x11, y, "Domaine public (GESDEC)")
614             y+=15
615 
616     print ("- technical caracteristics")
617     def caract_techn_function(PROFONDEUR_EQUIP, PROFONDEUR_MD, REFERENTIEL,
    DATE_NIVELLEMENT, TUBAGE, H_CREPINE, B_CREPINE, SONDE_ENREGISTREUSE):
```

```
618         global y
619         canvas.setFont('Helvetica-Bold', 10)
620         y=yCT
621         canvas.drawString(x2, y, "Caractéristiques techniques")
622         canvas.setFont('Helvetica', 10)
623         y-=20
624         canvas.setFont('Helvetica', 10)
625         canvas.drawString(x2, y,"Profondeur de l'équipement: ")
626 
627         if row[21] is not None:
628             PROFONDEUR_EQUIP=str("{:.2f}".format(float(PROFONDEUR_EQUIP)))
629             canvas.drawString(x22, y, PROFONDEUR_EQUIP + " m")
630         else:
631             canvas.drawString(x22, y, "-")
632         y-=15
633         canvas.drawString(x2, y,"Profondeur du sondage (MD): ")
634         if row[22] is not None:
635             PROFONDEUR_MD=str("{:.2f}".format(float(PROFONDEUR_MD)))
636             canvas.drawString(x22, y, PROFONDEUR_MD + " m")
637         else:
638             canvas.drawString(x22, y, "-")
639         y-=25
640         canvas.drawString(x2, y,"Nature Referentiel: ")
641         if row[19] is not None:
642             canvas.drawString(x22, y, REFERENTIEL)
643         else:
644             canvas.drawString(x22, y, "-")
645         y-=15
646         canvas.drawString(x2, y,"Date Nivellement: ")
647         if row[20] is not None:
648             canvas.drawString(x22, y, DATE_NIVELLEMENT)
649         else:
650             canvas.drawString(x22, y, "-")
651         y-=25
652         canvas.drawString(x2, y,"Tubage Matériel: ")
653         if row[23] is not None:
654             canvas.drawString(x22, y, TUBAGE)
655         else:
656             canvas.drawString(x22, y, "-")
657         y-=15
658         canvas.drawString(x2, y,"Crépine (haut et base): ")
659         if row[25] is not None and row[26] is not None:
660             H_CREPINE=str("{:.2f}".format(float(H_CREPINE)))
661             B_CREPINE=str("{:.2f}".format(float(B_CREPINE)))
662             canvas.drawString(x22, y, H_CREPINE + " - " + B_CREPINE + " m")
663         if row[25] is not None and row[26] is None:
664             H_CREPINE=str("{:.2f}".format(float(H_CREPINE)))
665             canvas.drawString(x22, y, H_CREPINE + " - " + "inconnu")
666         if row[25] is None and row[26] is not None:
667             B_CREPINE=str("{:.2f}".format(float(B_CREPINE)))
668             canvas.drawString(x22, y, "inconnu"+" - " + B_CREPINE + " m")
669         if row[25] is None and row[26] is None:
670             canvas.drawString(x22, y, "-")
671         y-=15
672         canvas.drawString(x2, y, "Sonde enregistreuse:")
673         if row[27] is not None:
674             canvas.drawString(x22, y, SONDE_ENREGISTREUSE)
675         else:
676             canvas.drawString(x22, y, "-")
677 
678     print ("- schema")
679     def schema_function(ALT_TERRAIN, ALT_REFERENTIEL, DIA_TUBE, DIA_FOND_TROU):
680         global y
681         if row[17] is not None and row[18]is not None:
682             diff = float(row[18]) - float(row[17]) #Calculating the difference between
            both altitudes
683             diff=str("{:.2f}".format(float(diff)))
684             # Formatting (two characters after the comma)
685             print(ALT_TERRAIN)
686             print (ALT_REFERENTIEL)
687             ALT_REFERENTIEL = str("{:.2f}".format(float(ALT_REFERENTIEL)))
688             ALT_TERRAIN = str("{:.2f}".format(float(ALT_TERRAIN)))
```

```
689             if ALT_REFERENTIEL <= ALT_TERRAIN:
690                 yI= yIT
691                 yII= yI + 243
692                 yIII= yII-22
693                 xI= xIT
694                 xII= xI-47
695                 canvas.drawImage(schema1, 315, yI, width=250, preserveAspectRatio=True,
                    mask ='auto')
696                 canvas.setFont('Helvetica', 8)
697                 canvas.drawString(xI, yII, "Alt. Terrain")
698                 yII-=15
699                 canvas.setFont('Helvetica-Bold', 10)
700                 canvas.drawString(xI, yII, str(ALT_TERRAIN) + " m")
701                 yII-=16
702                 canvas.setFont('Helvetica', 8)
703                 canvas.drawString(xI, yII, "Alt. Referentiel")
704                 yII-=15
705                 canvas.setFont('Helvetica-Bold', 10)
706                 canvas.drawString(xI, yII, str(ALT_REFERENTIEL) + " m")
707                 canvas.setFont('Helvetica', 8)
708                 canvas.drawString(xII, yIII, diff + " m")
709                 yII-=66
710                 canvas.setFont('Helvetica', 8)
711                 canvas.drawString(xII, yII, "Diamètre Tube")
712                 yII-=15
713                 canvas.setFont('Helvetica-Bold', 10)
714                 if row[24] is not None:
715                     DIA_TUBE=str("{:.2f}".format(float(DIA_TUBE)))
716                     canvas.drawString(xII, yII, DIA_TUBE + " \"")
717                 else:
718                     canvas.drawString(xII, yII, "-")
719                 canvas.setFont('Helvetica', 8)
720                 yII -= 4
721                 canvas.drawString(xI, yII, "Diamètre Forage")
722                 yII-=15
723                 canvas.setFont('Helvetica-Bold', 10)
724                 if row[16] is not None:
725                     DIA_FOND_TROU=str("{:.2f}".format(float(DIA_FOND_TROU)))
726                     canvas.drawString(xI, yII, DIA_FOND_TROU + " m")
727                 else:
728                     canvas.drawString(xI, yII, "-")
729
730             if ALT_TERRAIN < ALT_REFERENTIEL:
731                 yI=yIR
732                 yII= yI+218
733                 yIII=yII-22
734                 xI = xIR
735                 xII = xI-50
736                 canvas.drawImage(schema2, 315, yI, width=250, preserveAspectRatio=True,
                    mask ='auto')
737                 canvas.setFont('Helvetica', 8)
738                 canvas.drawString(xI, yII, "Alt. Referentiel")
739                 yII-=15
740                 canvas.setFont('Helvetica-Bold', 10)
741                 canvas.drawString(xI, yII, str(ALT_REFERENTIEL) + " m")
742                 canvas.setFont('Helvetica', 8)
743                 yII-=15
744                 canvas.drawString(xI, yII, "Alt. Terrain")
745                 yII-=15
746                 canvas.setFont('Helvetica-Bold', 10)
747                 canvas.drawString(xI, yII, str(ALT_TERRAIN) + " m")
748                 canvas.setFont('Helvetica', 8)
749                 canvas.drawString(xII, yIII, diff + " m")
750                 yII-=42
751                 canvas.setFont('Helvetica', 8)
752                 canvas.drawString(xII, yII, "Diamètre Tube")
753                 yII-=15
754                 canvas.setFont('Helvetica-Bold', 10)
755                 if row[24] is not None:
756                     DIA_TUBE=str("{:.2f}".format(float(DIA_TUBE)))
757                     canvas.drawString(xII, yII, DIA_TUBE + " \"")
758                 else:
```

```
                canvas.drawString(xII, yII, "-")
            canvas.setFont('Helvetica', 8)
            yII-=3
            canvas.drawString(xI, yII, "Diamètre Forage")
            canvas.setFont('Helvetica-Bold', 10)
            yII-=15
            if row[16] is not None:
                DIA_FOND_TROU=str("{:.2f}".format(float(DIA_FOND_TROU)))
                canvas.drawString(xI, yII, DIA_FOND_TROU + " m")
            else:
                canvas.drawString(xI, yII, "-")


    else: #Si altitude terrain ou altitude referentiel pas renseigné
        canvas.setFont('Helvetica', 10)
        y = yCT -20
        canvas.drawString(x1, y, "Alt. Terrain")
        if row[17] is not None:
            ALT_TERRAIN = str("{:.2f}".format(float(ALT_TERRAIN)))
            canvas.setFont('Helvetica', 10)
            canvas.drawString(x11, y, str(ALT_TERRAIN) + " m")
        else:
            canvas.drawString(x11, y, "-")
        y-=15
        canvas.setFont('Helvetica', 10)
        canvas.drawString(x1, y, "Alt. Referentiel")
        if row[18] is not None:
            ALT_REFERENTIEL = str("{:.2f}".format(float(ALT_REFERENTIEL)))
            canvas.setFont('Helvetica', 10)
            canvas.drawString(x11, y, str(ALT_REFERENTIEL) + " m")
        else:
            canvas.drawString(x11, y, "-")
        y-=25
        canvas.setFont('Helvetica', 10)
        canvas.drawString(x1, y, "Diamètre Tube")
        canvas.setFont('Helvetica', 10)
        if row[24] is not None:
            DIA_TUBE=str("{:.2f}".format(float(DIA_TUBE)))
            canvas.drawString(x11, y, DIA_TUBE + " \"")
        else:
            canvas.drawString(x11, y, "-")

print ("- joining map")
def addmap_function():
    arcpy.env.workspace = maps
    criteria3= "map" + str(ID_EQUIPEMENT) + ".png"
    mappicture = maps + criteria3
    PO3=Image.open(mappicture)
    canvas.drawImage(mappicture, dist, yM, width=widthH, preserveAspectRatio=True)

print ("- joining photos")
def addphotos_function():
    # Select Pictures
    global photoList1
    global photoList2
    global heightP1
    arcpy.env.workspace = photos
    criteria1= "*P" + ID_EQUIPEMENT + "_*1.jpg"
    photoList1=arcpy.ListFiles(criteria1)
    criteria2= "*P" + ID_EQUIPEMENT + "_*2.jpg"

    photoList2=arcpy.ListFiles(criteria2)


    # Photo 1
    if photoList1: #should be met when list is not empty
        photo1 = photos + "\\" + photoList1[0]
        PO1 = Image.open(photo1)
        # Check the orientation
        photo1W=PO1.width
        photo1H=PO1.height
        if photo1W > photo1H:
```

```
831                 orientation1 = "Landscape"
832                 widthP1 = widthH
833                 yR = yPL
834             else:
835                 orientation1 = "Portrait"
836                 widthP1 = widthH -50
837                 yR = yPP
838             canvas.drawImage(photo1, xP1, yP, width=widthP1, preserveAspectRatio=True)
839             heightP1=photo1H * widthP1/photo1W +1
840             canvas.setLineWidth(0.3)
841             canvas.setStrokeColorRGB(0.5, 0.5, 0.5, alpha =0.5)
842             canvas.rect(xP1, yR, widthP1, heightP1, stroke = 1, fill=0)
843 
844         # Photo 2
845         if photoList2:
846             photo2 = photos + "\\" + photoList2[0]
847             PO2= Image.open(photo2)
848             photo2W=PO2.width
849             #Check the orientation
850             photo2H=PO2.height
851             if photo2W > photo2H:
852                 orientation2 = "Landscape"
853                 widthP2 = widthH
854                 xP2= xP2L
855                 yR = yPL
856             else:
857                 orientation2 = "Portrait"
858                 widthP2=widthH-50
859                 xP2 = xP2P
860                 yR = yPP
861             canvas.drawImage(photo2, xP2, yP, width=widthP2, preserveAspectRatio=True)
862             heightP2=photo2H * widthP2/photo2W
863             canvas.setLineWidth(0.3)
864             canvas.setStrokeColorRGB(0.5, 0.5, 0.5, alpha =0.5)
865             canvas.rect(xP2, yR, widthP2, heightP2, stroke=1, fill=0)
866 
867     print("- mesures")
868     def measures_function(MAX_DATE_MESURE, MIN_Date, MAX_Date, COUNT_NIVEAU_MESURE,
        MEAN_NIVEAU_NAPPE, DER_NIVEAU_NAPPE, COUNT_TEMPERATURE, MEAN_TEMPERATURE):
869         global y
870         global yRessource
871         global yMesure
872         global yMesureTop
873         #Reecriture des dates
874         if row[40]is not None:
875             MAX_DATE_MESURE = datetime.strptime(MAX_DATE_MESURE, "%Y-%m-%d %H:%M:%S")
876             dayDER = MAX_DATE_MESURE.day
877             if dayDER < 10:
878                 dayDER = "0" + str(dayDER)
879             monthDER = MAX_DATE_MESURE.month
880             if monthDER <10:
881                 monthDER = "0" + str(monthDER)
882             DER_Date_f = str(dayDER) + "." + str(monthDER) + "." +
            str(MAX_DATE_MESURE.year)
883         if row[41] is not None:
884             MIN_Date = datetime.strptime(MIN_Date, "%Y-%m-%d %H:%M:%S")
885             dayMIN = MIN_Date.day
886             if dayMIN <10:
887                 dayMIN = "0" + str(dayMIN)
888             monthMIN = MIN_Date.month
889             if monthMIN <10:
890                 monthMIN = "0" + str(monthMIN)
891             MIN_Date_f = str(dayMIN) + "." + str(monthMIN) + "." + str(MIN_Date.year)
892         if row[42] is not None:
893             MAX_Date = datetime.strptime(MAX_Date, "%Y-%m-%d %H:%M:%S")
894             dayMAX = MAX_Date.day
895             if dayMAX < 10:
896                 dayMAX = "0"+ str(dayMAX)
897             monthMAX = MAX_Date.month
898             if monthMAX <10:
899                 monthMAX = "0" + str(monthMAX)
900             MAX_Date_f = str(dayMAX) + "." + str(monthMAX) + "." + str(MAX_Date.year)
```

```
901
902         #Definition du nombre de mesures
903         if row[35] is not None:
904             COUNT_NIVEAU_MESURE = str(int(float(COUNT_NIVEAU_MESURE)))
905             MesuresNappe = COUNT_NIVEAU_MESURE
906         else:
907             MesuresNappe = "0"
908
909         if row[36]is not None:
910             MEAN_NIVEAU_NAPPE = str("{:.2f}".format(float(MEAN_NIVEAU_NAPPE)))
911             Moyenne = str(MEAN_NIVEAU_NAPPE) + " m"
912         else:
913             Moyenne = "-"
914         if row[37] is not None:
915             Minimum = str(MIN_NIVEAU_NAPPE) + " m (" + str(MIN_Date_f) + ")"
916         else:
917             Minimum = "-"
918         if row[38] is not None:
919             Maximum = str(MAX_NIVEAU_NAPPE) + " m (" + str(MAX_Date_f) + ")"
920         else:
921             Maximum = "-"
922         if row[43] is not None:
923             DER_NIVEAU_NAPPE = str("{:.2f}".format(float(DER_NIVEAU_NAPPE)))
924             Derniere = str(DER_NIVEAU_NAPPE) + " m (" + str(DER_Date_f) + ")"
925             if row[43] == 0:
926                 Derniere = "Sec"
927         else:
928             Derniere = "-"
929
930         if row[44] is not None:
931             COUNT_TEMPERATURE = str(int(float(COUNT_TEMPERATURE)))
932             MesuresTemp = COUNT_TEMPERATURE
933             MEAN_TEMPERATURE = str("{:.1f}".format(float(MEAN_TEMPERATURE)))
934             MoyenneTemp = str(MEAN_TEMPERATURE) + "°C"
935         else:
936             MesuresTemp = "0"
937             MoyenneTemp = "-"
938
939         canvas.setFont('Helvetica-Bold', 10)
940         canvas.drawString(x1, y, "Mesures")
941         y-=20
942         canvas.setFont('Helvetica', 10)
943         canvas.drawString(x1, y, "Niveaux de nappe (" + str(MesuresNappe)+ " mesures)")
944
945         y-=20
946         canvas.setFont('Helvetica-Oblique', 9)
947         if row[35] is not None:
948             if int(row[35]) >2:    #at least two measures to determine mean and min, max
            and temperature, conditions can not be combined (if None fail)
949                 yM= y
950                 canvas.drawString(xM1, y, "Moyenne")
951                 canvas.drawString(xM2, y, Moyenne)
952                 y-=15
953                 canvas.drawString(xM1, y, "Maximum")
954                 canvas.drawString(xM2, y, Maximum)
955                 y-=15
956                 canvas.drawString(xM1, y, "Minimum")
957                 canvas.drawString(xM2, y, Minimum)
958                 y-=15
959                 yD=y
960                 canvas.drawString(xM1, y, "Dernière mesure")
961                 canvas.drawString(xM2, y, Derniere)
962                 y-=20
963
964                 # Draw Line to devide text
965                 canvas.setStrokeColorRGB(0.5, 0.5, 0.5, alpha =0.5)
966                 canvas.setLineWidth(0.5)
967                 canvas.line(x11-25, yD, x11-25, yM + 5)
968                 #Bring back black color for text (standard)
969                 canvas.setStrokeColorRGB(0, 0, 0, alpha =1)
970                 canvas.setFillColorRGB(0, 0, 0, alpha= 1)
971
```

```
        canvas.setFont('Helvetica', 10)

        canvas.drawString(x1, y, "Température de la nappe (" + str(MesuresTemp)+ "
        mesures)")
        canvas.setFont('Helvetica-Oblique', 9)
        if row[39] is not None:
            y-=20
            canvas.drawString(xM1, y, str("Moyenne"))
            canvas.drawString(xM2, y, MoyenneTemp)

            # Draw Line to devide text
            canvas.setStrokeColorRGB(0.5, 0.5, 0.5, alpha =0.5)
            canvas.line(x11-25, y + 10, x11-25, y-2)
            #Bring back black color for text (standard)
            canvas.setStrokeColorRGB(0, 0, 0, alpha =1)
            canvas.setFillColorRGB(0, 0, 0, alpha= 1)
        yT = y

        yMesure= y -10
        layout_boxesP1()


    print ("- ressources")
    def ressources_function():
        global y
        global yGraphique
        global yGraphiqueTop

        canvas.setFont('Helvetica-Bold', 10)
        canvas.drawString(x2, y,"Ressource ")
        y-=20
        canvas.setFont('Helvetica', 10)
        canvas.drawString(x2, y,"Nom de la nappe:")
        if row[28] is not None:
            x= len(NOM)
            if x <23:
                canvas.drawString(x22, y, NOM)
            else:
                split = NOM.rsplit('- ',2)
                n= len(split)
                first = split[0]
                if n <3:
                    second = split[1]
                if n ==3:
                    second = split[1] + " - " + split[2]
                if n >3:
                    second = split[1] + " - " + split[2] + " - " + split[3]
                canvas.drawString(x22, y, first+ "-")
                y-=15
                canvas.drawString(x22, y, second)

        else:
            canvas.drawString(x22, y, "-")
        y-=15
        canvas.setFont('Helvetica', 10)
        canvas.drawString(x2, y,"Type de nappe:")
        if row[29] is not None:
            canvas.drawString(x22, y,NAPPE)
        else:
            canvas.drawString(x22, y, "-")
        y-=15
        canvas.drawString(x2, y,"Pression de la nappe:")
        if row[30] is not None:
            canvas.drawString(x22, y,PRESSION_NAPPE)
        else:
            canvas.drawString(x22, y, "-")
        y-=15
        canvas.drawString(x2, y,"Unités géologiques:")
        if row[31] is not None and row[32] is None: #Necessary to recall rows (otherwise
        not working)
            UNITES = UNITE_GEOL_1
        elif row[32] is not None and row[31] is None:
```

```
            UNITES = UNITE_GEOL_2
        elif row[31] is not None and row[32] is not None:
            UNITES = str(UNITE_GEOL_1) + ", " + str(UNITE_GEOL_2)
        else:
           UNITES = "-"
        x = len(UNITES)
        if x <25:
            canvas.drawString(x22, y, UNITES)
        else:
            canvas.drawString(x22, y, UNITE_GEOL_1)
            y-=15
            canvas.drawString(x22, y, UNITE_GEOL_2)

        y -=102
        if row[33] == "Oui":
            yGraphique= y-22
            yGraphiqueTop= y + 30
            canvas.setFont('Helvetica', 10)
            canvas.drawString(x2, y,"Représentation graphique: ")
            canvas.setFont('Helvetica-Bold', 10)
            canvas.drawString(x22, y, "Niveau d'eau, de pluviométrie et températures du
            piézomètre")
            layout_boxesP3()
            canvas.linkURL(str(LIEN_VH), (25, yGraphique, 570, yGraphiqueTop), relative
            =1)
            y-=55


    print ("- releve geologique")
    def releve_geologique_function():
        global y
        global yReleveTop
        global yReleve
        canvas.setFont('Helvetica', 10)
        canvas.drawString(x2, y, "Représentation graphique: ")
        canvas.setFont('Helvetica-Bold', 10)
        canvas.drawString(x22, y, "Relevé géologique du sondage")
        condition =  str(ID_SONDAGE)+ ".pdf"
        arcpy.env.workspace = sondagesprive
        prive= arcpy.ListFiles(condition)
        if prive: #If empty it is considered false so it will pass to else
            y-=15
            yReleveTop = y + 45
            yReleve = y - 22
            canvas.setFont('Helvetica-Oblique', 8)
            releve = sondagesprive + str(ID_SONDAGE)+ ".pdf"
            canvas.drawString(x22, y, str(releve))
        else: #Relevé is public
            yReleveTop = y + 30
            yReleve = y - 22
            releve = sondagespublic + str(ID_SONDAGE) + ".pdf"
            canvas.linkURL(str(releve), (25, yReleve, 570, yReleveTop), relative =1)
        layout_boxesP4()


    # Layout Functions
    print("-------------------------------------------------------------------------")
    print("Definition of the layout")
    print("-------------------------------------------------------------------------")
    def neatline():
        canvas.setLineWidth(0.3)
        canvas.setStrokeColorRGB(0.5, 0.5, 0.5, alpha =1)
        canvas.rect(15, 15, 565, 815, stroke = 1, fill = 0)

    def layout_boxesP0():
        #Set Box around Localisation and access
        xbox1=x1-10
        heightT = yS1 -ybox1 + 13
        canvas.setLineWidth(0.3)
        canvas.setFillColorRGB(0.1, 0.1, 0.1, alpha = 0.08)
        canvas.rect(xbox1, ybox1, widthH, heightT, stroke = 0, fill = 1)
        #Bring back black color for text (standard)
```

```
1112        canvas.setStrokeColorRGB(0, 0, 0, alpha =1)
1113        canvas.setFillColorRGB(0, 0, 0, alpha= 1)
1114
1115    def layout_boxesP1():
1116        xbox1=x1-10
1117        #Set Box around Measures
1118        canvas.setFillColorRGB(0.1, 0.1, 0.1, alpha = 0.08)
1119        # Draw a rectangle
1120        canvas.setLineWidth(0.5)
1121        canvas.setStrokeColorRGB(0,0,0, alpha=1)
1122        canvas.setStrokeColorRGB(0,0,0, alpha=1)
1123        canvas.rect(xbox1, yMesure, widthH, yMesureTop-yMesure+7, stroke = 1, fill = 0)
1124        #Bring back black color for text (standard)
1125        canvas.setStrokeColorRGB(0, 0, 0, alpha =1)
1126        canvas.setFillColorRGB(0, 0, 0, alpha= 1)
1127
1128
1129    def layout_boxesP2():
1130        # Set Box around Photos (grey box)
1131        canvas.setLineWidth(0.5)
1132        canvas.setFillColorRGB(0.1, 0.1, 0.1, alpha = 0.08)
1133        canvas.rect(dist, yPP, widthF, heightP1, stroke=0, fill=1)
1134        #Bring back black color for text (standard)
1135        canvas.setStrokeColorRGB(0, 0, 0, alpha =1)
1136        canvas.setFillColorRGB(0, 0, 0, alpha= 1)
1137
1138    def line_function():
1139        canvas.setLineWidth(0.8)
1140        canvas.setStrokeColorRGB(0.1, 0.1, 0.1, alpha =0.15)
1141        canvas.setFillColorRGB(0.1, 0.1, 0.1, alpha = 0.08)
1142        y = yR+ 25
1143        canvas.line(dist, y, x2 +widthF-5, y)
1144        #Bring back black color for text (standard)
1145        canvas.setStrokeColorRGB(0, 0, 0, alpha =1)
1146        canvas.setFillColorRGB(0, 0, 0, alpha= 1)
1147
1148    def layout_boxesP3():
1149        #Set Box around Représentation graphique du niveau d'eau, de pluviométrie et
        températures du piézomètre
1150        canvas.setFillColorRGB(0.1, 0.1, 0.1, alpha = 0.08)
1151        # Draw a rectangle
1152        canvas.rect(dist, yGraphique, widthF, yGraphiqueTop-yGraphique-5, stroke = 0,
        fill = 1)
1153        # Draw Line to underline text
1154        canvas.setFillColorRGB(0, 0, 0)
1155        canvas.line(x22,yGraphiqueTop-33, x22 +290, yGraphiqueTop-33)
1156        #Bring back black color for text (standard)
1157        canvas.setStrokeColorRGB(0, 0, 0, alpha =1)
1158        canvas.setFillColorRGB(0, 0, 0, alpha= 1)
1159
1160    def layout_boxesP4():
1161        #Set Box around Relevé géologique du sondage
1162        canvas.setFillColorRGB(0.1, 0.1, 0.1, alpha = 0.08)
1163        # Draw a rectangle
1164        rect = canvas.rect(dist, yReleve, widthF, yReleveTop-yReleve-5, stroke = 0, fill
        = 1)
1165        # Draw Line to underline text
1166        canvas.setFillColorRGB(0, 0, 0)
1167        canvas.line(x22,yReleveTop -33, x22 +148, yReleveTop -33)
1168        #Bring back black color for text (standard)
1169        canvas.setStrokeColorRGB(0, 0, 0, alpha =1)
1170        canvas.setFillColorRGB(0, 0, 0, alpha= 1)
1171
1172    def update():
1173        #Import date
1174        canvas.setStrokeColorRGB(0.1, 0.1, 0.1, alpha =0.4)
1175        canvas.setFont('Helvetica-Oblique', 7)
1176        today = date.today()
1177        day = today.day
1178        month = today.month
1179        year = today.year
1180        canvas.drawString(x11+25, 17, "Dernière mise à jour : " + str(day) + "." +
```

```
        str(month) + "." + str(year))
1181
1182
1183    ### LOOP THROUGH ALL EXISTING PIEZOMETRES TO CREATE REPORT
1184    print("--------------------------------------------------------------------------------")
1185    print("Creation of the reports in loop:")
1186    print("--------------------------------------------------------------------------------")
1187    for i in piezos:
1188        expression = '"ID_EQUIPEMENT" = ' + str(i)
1189        print("For piezometre: " + expression)
1190        # Grasp the data for each piezometre
1191        print("- extraction of the individual data")
1192        with arcpy.da.SearchCursor(fc, fields, expression) as cursor:
1193            for row in cursor:
1194                ID_EQUIPEMENT = str(int(float(row[0])))
1195                ID_SONDAGE = str(int(float(row[1])))
1196                NOM_PIEZO_TEMP =str(row[2])
1197                ID_TEMP_SP = str(row[3])
1198                COORD_X =str(row[4])
1199                COORD_Y=str(row[5])
1200                RUE=str(row[6])
1201                NPA=row[7]
1202                COMMUNE=str(row[8])
1203                DISTANCE_A_PIEDS=str(row[9])
1204                CONDITION_ACCES=str(row[10])
1205                OUVERTURE=str(row[11])
1206                ETAT_EQU = str(row[12])
1207                USAGE = str(row[13])
1208                TOURNEE= str(row[14])
1209                FIN_TRAVAUX = row[15]           #no string possible as formatting
                    required for the places after the comma
1210                DIA_FOND_TROU = row[16]
1211                ALT_TERRAIN = row[17]           #no string possible as formatting
                    required for the places after the comma
1212                ALT_REFERENTIEL = row[18]       #no string possible as formatting
                    required for the places after the comma
1213                REFERENTIEL = str(row[19])
1214                DATE_NIVELLEMENT = str(row[20])
1215                PROFONDEUR_EQUIP = row[21]      #no string possible as formatting
                    required for the places after the comma
1216                PROFONDEUR_MD = row[22]         #no string possible as formatting
                    required for the places after the comma
1217                TUBAGE = str(row[23])
1218                DIA_TUBE= row[24]               #no string possible as formatting
                    required for the places after the comma
1219                H_CREPINE = row[25]
1220                B_CREPINE=row[26]
1221                SONDE_ENREGISTREUSE = str(row[27])
1222                NOM = str(row[28])
1223                NAPPE = str(row[29])
1224                PRESSION_NAPPE = str(row[30])
1225                UNITE_GEOL_1 = str(row[31])
1226                UNITE_GEOL_2 = str(row[32])
1227                AUTOMATIQUE =str(row[33])
1228                LIEN_VH = str(row[34])
1229                COUNT_NIVEAU_MESURE = row[35]
1230                MEAN_NIVEAU_NAPPE = row[36]     #no string possible as formatting
                    required for the places after the comma
1231                MIN_NIVEAU_NAPPE = row[37]
1232                MAX_NIVEAU_NAPPE = row[38]
1233                MEAN_TEMPERATURE = row[39]
1234                MAX_DATE_MESURE = row[40]
1235                MIN_Date = row[41]
1236                MAX_Date = row[42]
1237                DER_NIVEAU_NAPPE = row[43]
1238                COUNT_TEMPERATURE=row[44]
1239
1240
1241        ### IMPORT LIBRARIES
1242        # Import reportlab
1243        from reportlab.pdfgen import canvas
1244        from reportlab.lib.pagesizes import A4, portrait
```

```
1245        from reportlab.lib import colors
1246        from reportlab import platypus
1247        from reportlab.lib.styles import ParagraphStyle as PS
1248        from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph
1249
1250        #Import PIL
1251        from PIL import Image
1252
1253        #Import date
1254        from datetime import date
1255
1256        ### DEFINE THE CANVAS COMPOSITION
1257        print("- definition of the canvas caracteristics")
1258        # Define name of the canvas
1259        pdfname = outputpath + "FichePiezo_" +str(ID_SONDAGE)+ "_P" + str(ID_EQUIPEMENT)
            + ".pdf"
1260
1261        # Define canvas orientation
1262        canvas=canvas.Canvas(pdfname,pagesize=portrait(A4))
1263        canvas.width, canvas.height = A4
1264
1265        # Define positions
1266        define_positions()
1267
1268        # Define Title
1269        title_function(y)
1270
1271        # Define Neatline
1272        neatline()
1273
1274        # Add MAPS
1275        print("- creation of the map and inclusion on the canvas")
1276        create_map(ID_EQUIPEMENT)
1277        addmap_function()
1278
1279        # Add Schema
1280        print("- creation of the schema and inclusion on the canvas")
1281        schema_function(ALT_TERRAIN, ALT_REFERENTIEL, DIA_TUBE, DIA_FOND_TROU)
1282
1283        # Localisation
1284        print("- add information on localisation")
1285        localisation_function(NPA)
1286
1287        # Access
1288        print("- add information on access")
1289        access_function()
1290
1291        # Activity
1292        print("- add information on activity")
1293        activity_function()
1294
1295        # Caractéristiques techniques
1296        print("- add information on technical caracteristics")
1297        caract_techn_function(PROFONDEUR_EQUIP, PROFONDEUR_MD, REFERENTIEL,
            DATE_NIVELLEMENT, TUBAGE, H_CREPINE, B_CREPINE, SONDE_ENREGISTREUSE)
1298
1299        # Add PHOTOS
1300        print("- add photos for this specific piezometre (if available)")
1301        addphotos_function()
1302
1303        # Define Layout Boxes for Page 1 (depending on the elements available for this
            piezometre)
1304        layout_boxesP0()
1305        if not photoList1 and not photoList2:
1306            PresencePhoto = 0
1307        else:
1308            PresencePhoto = 1
1309            layout_boxesP2()
1310
1311        # Insert Resources if no photos are available
1312        if PresencePhoto==0:
1313             # ADD INFORMATION ABOUT MEASURES
```

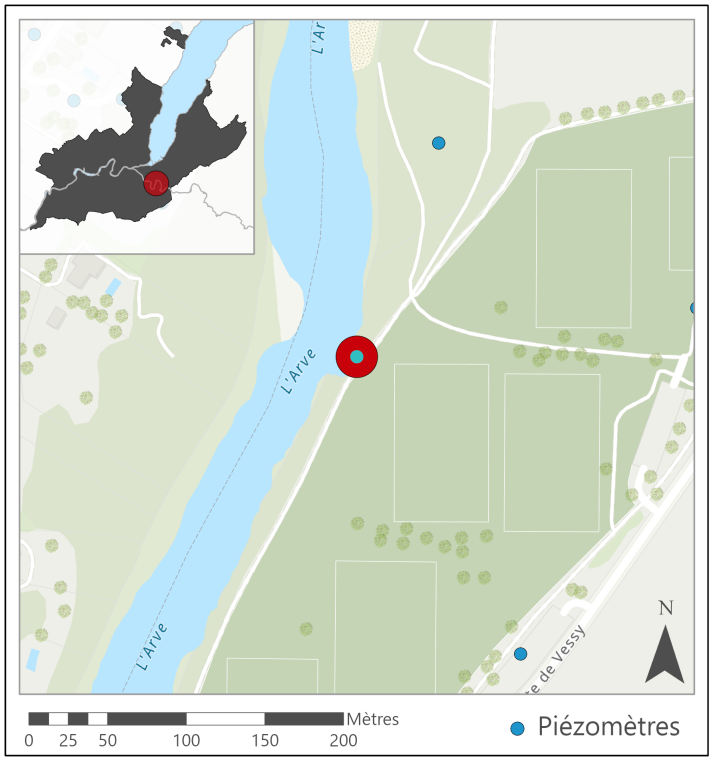
```
            print("- add information on measures")
            y = yR
            yMesureTop= yR + 10
            measures_function(MAX_DATE_MESURE, MIN_Date, MAX_Date, COUNT_NIVEAU_MESURE,
            MEAN_NIVEAU_NAPPE, DER_NIVEAU_NAPPE, COUNT_TEMPERATURE, MEAN_TEMPERATURE)

            # ADD INFORMATION ABOUT RESSOURCES
            print("- add information on resources")
            y = yR
            ressources_function()

            line_function()

            # ADD Link to LOG
            print("- add link to LOG")
            releve_geologique_function()

            #Add Date
            update()

            # Save PDF
            canvas.save()


        # Add another page if photos are available
        elif PresencePhoto == 1: #si photos sont présentes, une deuxième page est rajoutée

            #RAJOUT SECONDE PAGE
            canvas.showPage()
            define_positions()

            #Define Title
            title_function(y)

            # Define Neatline
            neatline()

            # ADD INFORMATION ABOUT MEASURES
            print("- add information on measures")
            y = yS2
            yMesureTop= yS2 + 7
            measures_function(MAX_DATE_MESURE, MIN_Date, MAX_Date, COUNT_NIVEAU_MESURE,
            MEAN_NIVEAU_NAPPE, DER_NIVEAU_NAPPE, COUNT_TEMPERATURE, MEAN_TEMPERATURE)


            # ADD INFORMATION ABOUT RESSOURCES
            print("- add information on resources")
            y=yS2
            ressources_function()

            # ADD Link to LOG
            print("- add link to LOG")
            releve_geologique_function()

            #Add Date
            update()

            #Save PDF
            canvas.save()

        print ("PDF successfully saved")

        del canvas


        print("--------------------------------------------------------------------------------
        ")

    #Delete Layer from the map
    m.removeLayer(lyr)

    #Script finished
```

```
1382   print ("Script has successfully run")
1383
```

Piézomètre ID_EQUIPEMENT 61

Vessy piézo 6

Sondage 1664



Localisation

Coord. X [CH1903+]:	2501484
Coord. Y [CH1903+]:	1115600
Adresse:	1234 Veyrier

Accès

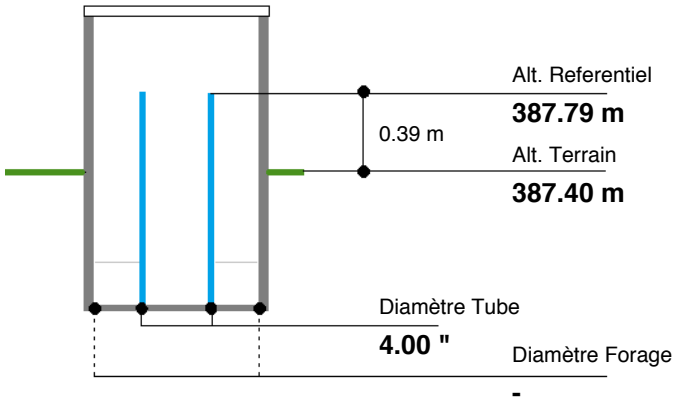
Distance à pieds:	-
Condition d'accès:	-
Ouverture:	-

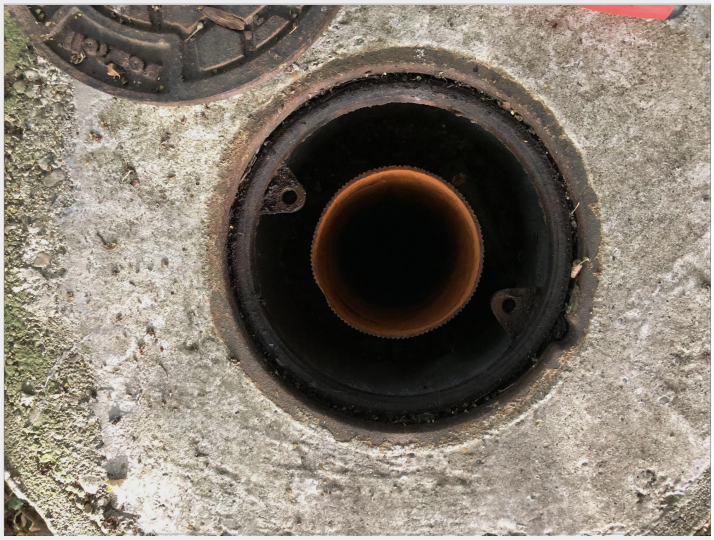
Activité

Etat de l'ouvrage:	Exploité / En activité
Réalisation du sondage:	01.01.1968
Usage:	Puits d'observation
Tournée (GESDEC):	Vessy
Contact:	Domaine public (GESDEC)

Caractéristiques techniques

Profondeur de l'équipement:	20.00 m
Profondeur du sondage (MD):	20.00 m
Nature Referentiel:	Tube piézométrique
Date Nivellement:	-
Tubage Matériel:	Inconnu
Crépine (haut et base):	15.00 - 20.00 m
Sonde enregistreuse:	Inconnu





REPUBLIQUE ET CANTON DE GENEVE

Piézomètre ID_EQUIPEMENT 61

Vessy piézo 6

Sondage 1664

Ressource

Nom de la nappe:	Genevois
Type de nappe:	Nappe principale
Pression de la nappe:	Libre
Unités géologiques:	Alluvion ancienne

Mesures

Niveaux de nappe (770 mesures)

Moyenne	*375.03 m*
Maximum	*377.12 m (29.02.2016)*
Minimum	*372.97 m (04.12.2006)*
Dernière mesure	*375.17 m (01.11.2021)*

Température de la nappe (34 mesures)

Moyenne	*9.6°C*

Représentation graphique: **Relevé géologique du sondage**

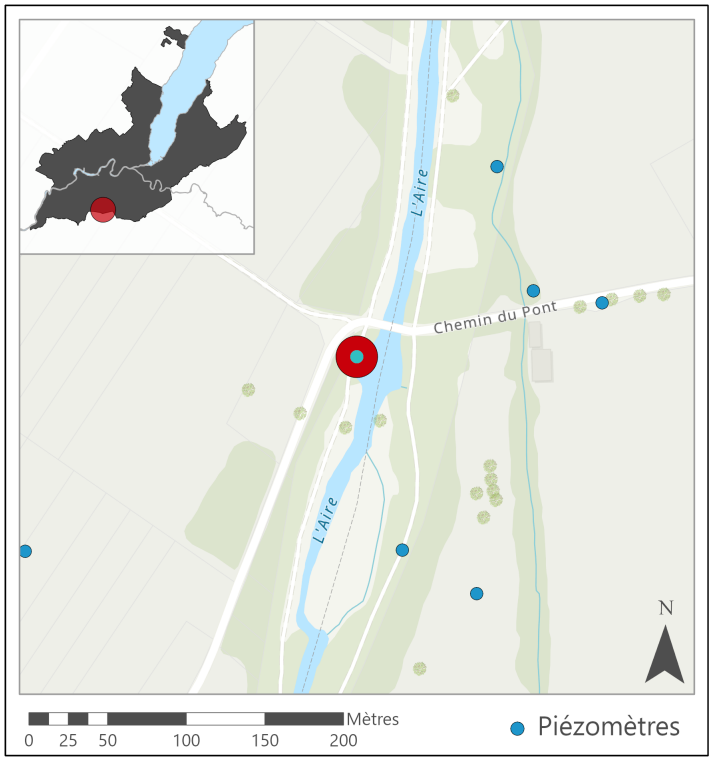
Dernière mise à jour : 07.01.2022

REPUBLIQUE ET CANTON DE GENEVE

Piézomètre ID_EQUIPEMENT 582

Perly-Certoux Nouveau

Sondage 17521



Localisation

Coord. X [CH1903+]:	2494741
Coord. Y [CH1903+]:	1112218
Adresse:	Route de Thérens 1233 Bernex

Accès

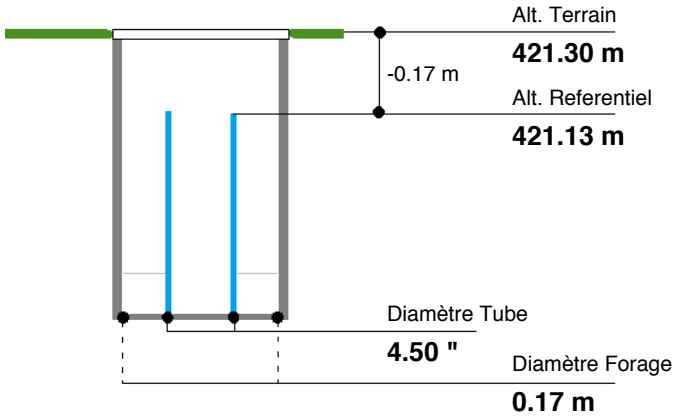
Distance à pieds:	20
Condition d'accès:	Accès libre
Ouverture:	clef imbus 8

Activité

Etat de l'ouvrage:	Exploité / En activité
Réalisation du sondage:	09.11.2018
Usage:	Puits d'observation
Tournée (GESDEC):	Genevois
Contact:	Domaine public (GESDEC)

Caractéristiques techniques

Profondeur de l'équipement:	96.00 m
Profondeur du sondage (MD):	96.00 m
Nature Referentiel:	Tube piézométrique
Date Nivellement:	-
Tubage Matériel:	PVC
Crépine (haut et base):	48.00 - 92.00 m
Sonde enregistreuse:	Non



Piézomètre ID_EQUIPEMENT 582

Perly-Certoux Nouveau

Sondage 17521

Ressource

Nom de la nappe:	Genevois
Type de nappe:	Nappe principale
Pression de la nappe:	En charge
Unités géologiques:	Alluvion ancienne

Mesures

Niveaux de nappe (89 mesures)

Moyenne	*373.59 m*
Maximum	*374.33 m (11.06.2021)*
Minimum	*372.73 m (20.12.2018)*
Dernière mesure	*374.33 m (11.06.2021)*

Température de la nappe (18 mesures)

Moyenne	*11.7°C*

Représentation graphique: **Niveau d'eau, de pluviométrie et températures du piézomètre**

Représentation graphique: **Relevé géologique du sondage**

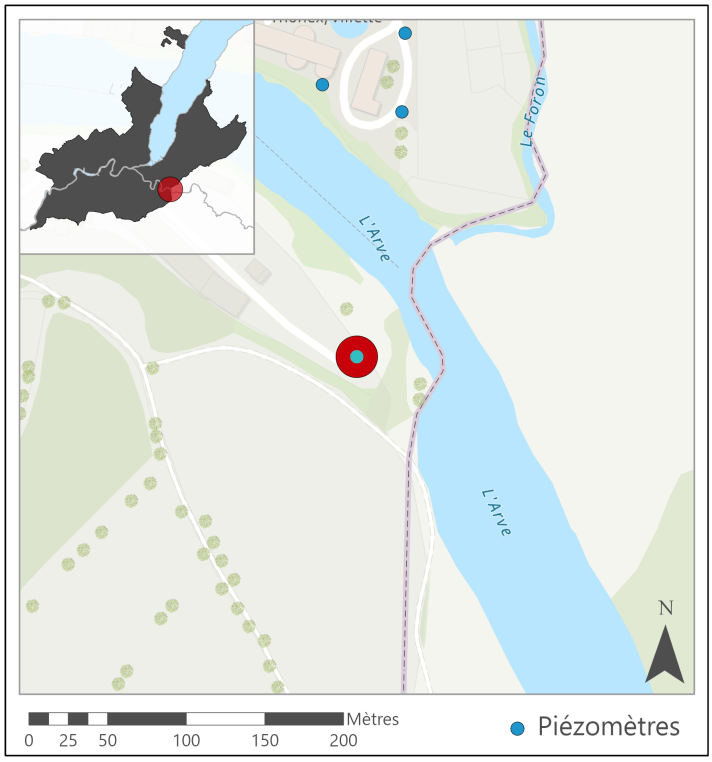
Dernière mise à jour : 07.01.2022

REPUBLIQUE ET CANTON DE GENEVE

Piézomètre ID_EQUIPEMENT 63

Maulini

Sondage 1723



Localisation

Coord. X [CH1903+]:	2503259
Coord. Y [CH1903+]:	1114812
Adresse:	1234 Veyrier

Accès

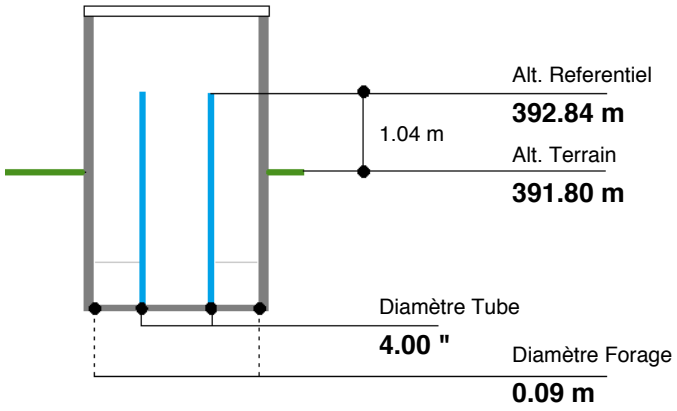
Distance à pieds:	-
Condition d'accès:	-
Ouverture:	-

Activité

Etat de l'ouvrage:	Exploité / En activité
Réalisation du sondage:	01.01.1969
Usage:	Puits d'observation
Tournée (GESDEC):	Genevois
Contact:	Domaine public (GESDEC)

Caractéristiques techniques

Profondeur de l'équipement:	27.60 m
Profondeur du sondage (MD):	40.00 m
Nature Referentiel:	Tube piézométrique
Date Nivellement:	-
Tubage Matériel:	PVC
Crépine (haut et base):	24.60 - 27.60 m
Sonde enregistreuse:	Inconnu



Ressource

Nom de la nappe:	Genevois
Type de nappe:	Nappe principale
Pression de la nappe:	Libre
Unités géologiques:	Alluvion ancienne

Mesures

Niveaux de nappe (845 mesures)

Moyenne	*375.16 m*
Maximum	*378.08 m (08.07.2016)*
Minimum	*372.14 m (19.09.2003)*
Dernière mesure	*376.05 m (08.10.2021)*

Température de la nappe (30 mesures)

Moyenne	*10.2°C*

Représentation graphique: **Niveau d'eau, de pluviométrie et températures du piézomètre**

Représentation graphique: **Relevé géologique du sondage**

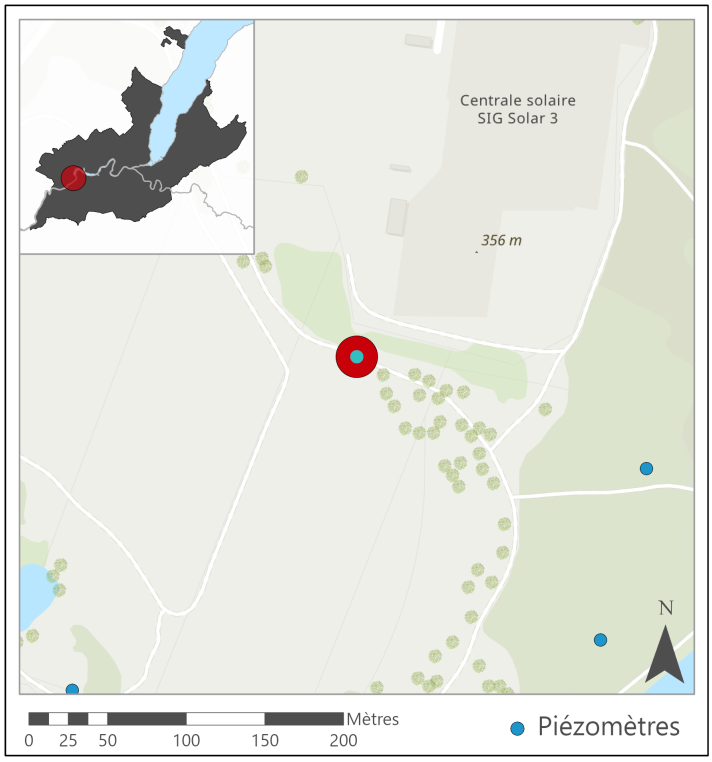
Dernière mise à jour : 07.01.2022

REPUBLIQUE ET CANTON DE GENEVE

Piézomètre ID_EQUIPEMENT 71

Verbois - S1

Sondage 2434



Localisation

Coord. X [CH1903+]:	2490974
Coord. Y [CH1903+]:	1116224
Adresse:	1281 Russin

Accès

Distance à pieds:	-
Condition d'accès:	-
Ouverture:	-

Activité

Etat de l'ouvrage:	Inconnu
Réalisation du sondage:	01.01.1972
Usage:	Puits d'observation
Tournée (GESDEC):	Inactif
Contact:	Domaine public (GESDEC)

Caractéristiques techniques

Profondeur de l'équipement:	-	Alt. Terrain	354.40 m
Profondeur du sondage (MD):	44.00 m	Alt. Referentiel	-
Nature Referentiel:	Inconnu	Diamètre Tube	-
Date Nivellement:	-		
Tubage Matériel:	-		
Crépine (haut et base):	-		
Sonde enregistreuse:	-		

Ressource

Nom de la nappe:	Rhône Aire
Type de nappe:	Nappe principale
Pression de la nappe:	Libre
Unités géologiques:	-

Mesures

Niveaux de nappe (0 mesures)

Température de la nappe (0 mesures)

Représentation graphique: **Relevé géologique du sondage**

Dernière mise à jour : 07.01.2022

Manuel: Fiches de piézomètre

Les fiches sont gérées et actualisées par Stéphanie Favre (pour le moment depuis son poste).

Avant l'exécution du script

Des droits particuliers doivent être accordés par la DIT à toute autre personne qui souhaite faire tourner le script (notamment l'accès de l'utilisateur à la basemap modifiée sur portal ainsi que l'extension reportLab doit être activée). La personne doit disposer d'une licence ArcGIS Pro.

- Basemap modifiée: présente dans le projet Piezometre_Map.aprx mais enregistrée sur Portal. Présente dans le projet Piezometres_Map qui est utilisé dans le script (*S:\UO5196\80_SYS_INFORMATION\SI_GEOLOGIE\05_DGSI\01_Applications_Metiers\SI_Geologie\Stages\Fiche_identite_piezo\01_FichesPIEZO\06_ProjetsArcGIS\Piezometres_Map*). Il faut avoir les droits d'accès à la couche afin d'exporter avec succès des cartes d'emplacement.

- Installation de l'extension reportLab: Nécessité de passer par une clé USB wifi pour installer cette extension inclue dans ArcGIS Pro. Pour rajouter le "package" ReportLab, il est requis de se diriger dans le menu Project de ArcGIS Pro, ensuite sur Python et de sélectionner l'option "Add Packages", où ReportLab peut être rajouté. D'office, il est requis d'effectuer un clone de l'environnement de projet d'ArcGIS Pro (requiert des droits d'administrateur). Dès que l'installation de ReportLab a abouti, le package devrait se trouver dans la liste des "Installed packages".

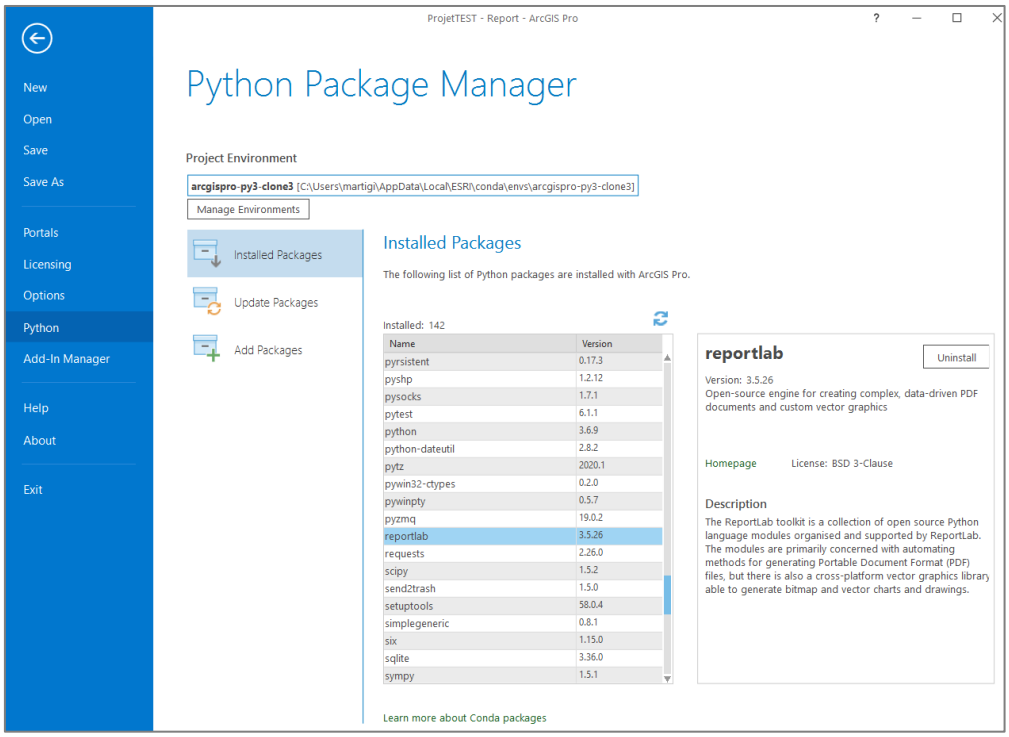


Figure 1: Installation du package ReportLab requis pour la création de rapports PDF sous Python, utilisant les données présents dans ArcGIS Pro

- Pour tester si reportLab a bien été installé et la génération de fiches est possible, vous pouvez faire tourner le script "TestReportLab.py" qui se trouve à l'emplacement suivant: *S:\UO5196\80_SYS_INFORMATION\SI_GEOLOGIE\05_DGSI\01_Applications_Metiers\SI_Geologie\Stages\Fiche_identite_piezo\01_FichesPIEZO\04_Scripts*

Précisions concernant les différents scripts

CreationFichePiezo_Serveur.py	Script généralisé pour actualiser et recréer toutes les fiches. Complètement basé sur SOLSTISS sur V: et le serveur.
CreationFichePiezo_Weekly_Serveur.py	Script permettant de créer les fiches pour des piézomètres récemment rajoutés à la base de données et pour lesquels la fiche n'était pas encore disponible. Complètement basé sur SOLSTISS sur V: et le serveur.
CreationFichePiezo_SFA.py	Script généralisé pour actualiser et recréer toutes les fiches. Basé sur SOLSTISS sur V:, le serveur ainsi que les projets ArcGIS sur le disque local (C:) de Stéphanie Favre (en local: but d'accélerer le processus).
CreationFichePiezo_Weekly_Piezo.py	Script permettant de créer les fiches pour des piézomètres récemment rajoutés à la base de données et pour lesquels la fiche n'était pas encore disponible. Basé sur SOLSTISS sur V:, le serveur ainsi que les projets ArcGIS sur le disque local (C:) de Stéphanie Favre (en local: but d'accélerer le processus).
ResizeImage.py	Script qui permet d'ajuster la taille de photos et de les enregistrer dans le bon format afin de permettre l'inclusion dans les fiches. La marche à suivre est décrite ci-dessous.
TestReportLab.py	Script qui permet de tester la bonne installation de reportLab.

L'exécution du script Python pour créer de nouvelles fiches pour tous les équipements

Le script se trouve à l'emplacement suivant:
S:\UO5196\80_SYS_INFORMATION\SI_GEOLOGIE\05_DGSI\01_Applications_Metiers\SI_Geologie\Stages\Fiche_identite_piezo\01_FichesPIEZO\04_Scripts

Pour exécuter manuellement le script (cf. Figure 2), faites un clic droit sur le script CreationFichePiezo_Serveur.py (Version partculière pour Stéphanie: _SFA) et sélectionnez "Edit with IDLE (ArcGIS Pro)". Après l'ouverture, sélectionnez "Run" dans le menu, puis "Run module". La génération des fiches peut durer plusieurs heures (idéalement faire tourner le script pendant la nuit).

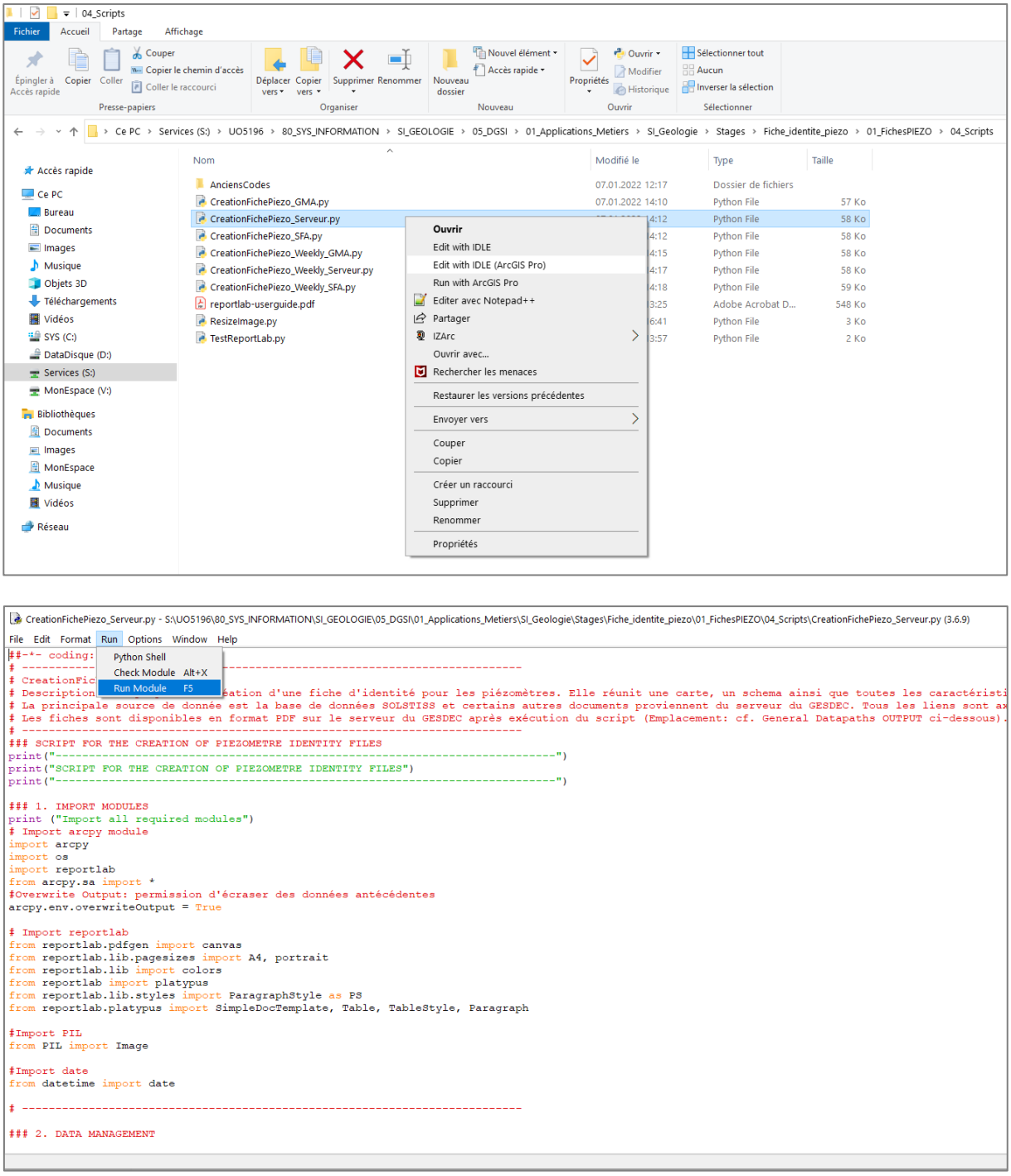


Figure 2: Illustration des étapes pour lancer le script Python

Les fiches créées se trouvent à l'emplacement suivant: *S:\UO5196\80_SYS_INFORMATION\SI_GEOLOGIE\05_DGSI\01_Applications_Metiers\SI_Geologie\Stages\Fiche_identite_piezo\01_FichesPIEZO\05_OutputFiches*

Les fiches suivent la logique suivante pour leurs noms: "FichePiezo_", ensuite le numéro de sondage (SCG) et ensuite "_P" et le numéro d'équipement du piézomètre. Si l'on souhaite chercher par numéro SCG (XXX) il faut insérer dans la recherche "_XXX_" pour trouver la bonne fiche et avec le numéro d'équipement (YYY) "_PYYY" avec les numéros respectifs associées (p.ex. la recherche "_5824_" et _P140" permettent de trouver la fiche du piézomètre dans la figure 3.).

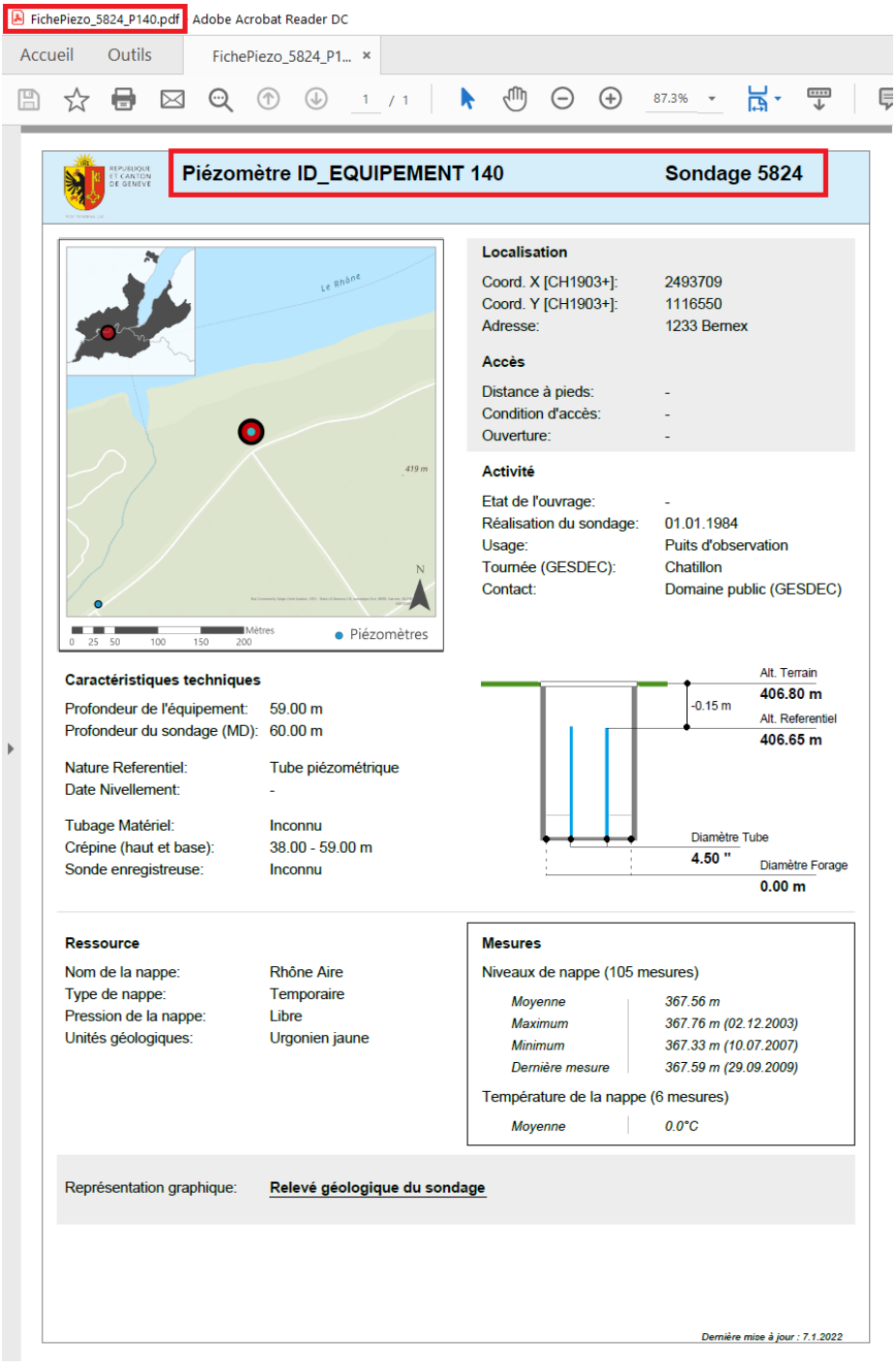


Figure 3: Illustration de la logique de recherche pour les fiches

5. Effacer les photos dans le dossier "Original" afin d'éviter que ces dernières soient retransformées à nouveau la prochaine fois.

"Entretien" du Projet Piezometre_Map.aprx

Le projet ArcGIS Pro est requis pour la création des cartes d'emplacement des piézomètres sur les fiches. En cas d'échec de finalisation du script, les couches de piezos ne sont pas automatiquement retirées du projet. Il est donc nécessaire de manuellement effacer la couche ou les couches "Piézomètre" du projet ArcGIS Pro Piezometres_Map.aprx. Il ne devrait y avoir aucune couche de piézomètre dans le projet et dans la carte "Map", uniquement la basemap "World Topographic Map" modifiée devrait y figurer. Il se peut que le script plante s'il y a trop de couches de piézomètres qui n'ont jamais été effacées.

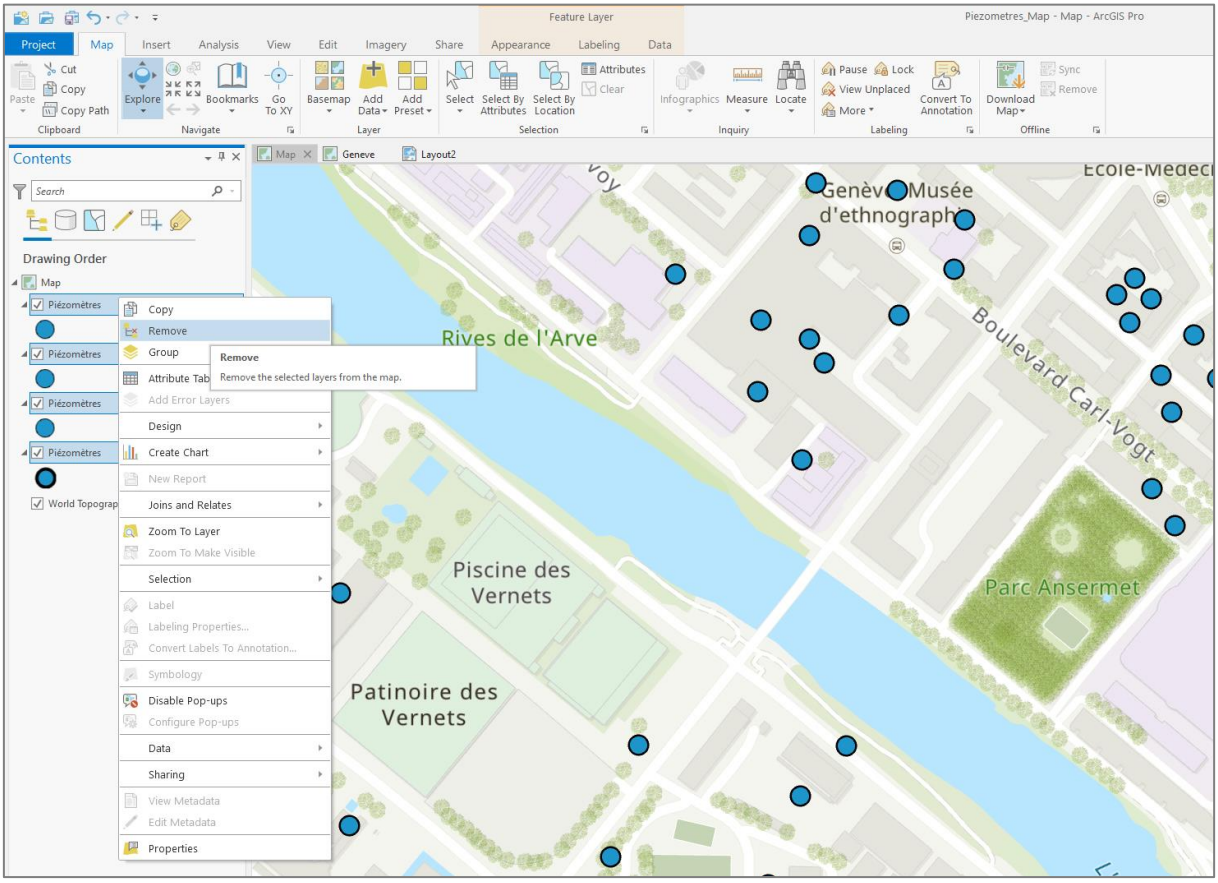


Figure 5: Représentation du projet Piezometres_Map et des démarches à affectuer pour effacer les couches de piézomètres.

Rajout de photos

Cette procédure explique comment intégrer les photos à la base de données utilisée pour les fiches d'identité de piézomètres. Cette procédure peut également être suivie pour remplacer et donc actualiser des anciennes photos (les anciennes seront écrasées). Les noms des photos indiquent le numéro d'équipement (ID_EQUIPEMENT) ainsi que la perspective (1 = à distance, 2= proche).

1. Copier –coller les photos dans le fichier "Original"

2. Renommer manuellement toutes les photos afin de correspondre au format suivant:
   Photo éloignée: "P" + ID_EQUIPEMENT + "_1.jpg"
   Photo de proche: "P" + ID_EQUIPEMENT + "_2.jpg"

   Exemples: Piézomètre 351 (correspondant à l'ID_EQUIPEMENT) ainsi que la perspective (1 = à distance, 2= proche) (cf. figure 4)

P351_1 *P351_2*

Figure 4: Exemple des noms attribués pour le piézomètre 351. Sur la gauche P351_1 et sur la droite P351_2

3. Ouvrir le script "ResizeImage.py" qui se trouve sur le serveur à l'emplacement suivant: *S:\UO5196\80_SYS_INFORMATION\SI_GEOLOGIE\05_DGSI\01_Applications_Metiers\SI_Geologie\Stages\Fiche_identite_piezo\01_FichesPIEZO\04_Scripts*

   Pour l'ouverture du script, clic droit sur le script et sélectionnez "Edit with IDLE (ArcGIS Pro)". Ensuite sélectionnez "Run" dans le menu, puis "Run module".

   Après la finalisation de la transformation, le message "The size of all photos has been successfully adjusted" s'affiche.

4. Les photos ajustées sont à partir de maintenant intégrés aux fiches dès la prochaine actualisation de ces dernières. Les photos ajustées se trouvent à l'emplacement suivant: *S:\UO5196\80_SYS_INFORMATION\SI_GEOLOGIE\05_DGSI\01_Applications_Metiers\SI_Geologie\Stages\Fiche_identite_piezo\01_FichesPIEZO\02_RepertoirePhotos\Resized*