

DATAVERSE



Déglise Camille – FIN2

ETML- Lausanne – Site de Vennes

13 mai 2024 – 03 juin 2024

Chef de projet : Sahli Bertrand

Experts : Borboën Nicolas & Favre Raphaël

Résumé

La première partie du présent rapport est une phase analytique de la situation de départ du projet. Initialement, il y a une détermination des spécificités du projet en lien avec le cahier des charges, les points techniques évalués et une planification initiale. Dans un deuxième temps, l'analyse complète du projet avec un SWOT, un diagramme de flux, l'analyse de la base de données, la conception de tests et enfin une planification détaillée.

La deuxième partie concerne la mise en œuvre à proprement parler. Les outils de travail comme Laravel ou TailwindCss sont présentés tout comme l'environnement de développement. Puis les neuf étapes de réalisation sont décrites avec leurs points clés ainsi qu'un tableau récapitulatif des différentes modifications majeures qui ont été apportées au fur à mesure de la réalisation.

Enfin en troisième partie, les résultats sont décrits avec un tableau de tests, reprenant la conception, une description des tests non-effectués, échoués, des bugs encore présents. Les différents bilans en fin de rapport décrivent les points à améliorer, les aspects positifs, négatifs des fonctionnalités, de la planification et au niveau personnel.

Table des matières

1	Spécifications.....	5
1.1	Titre	5
1.2	Description.....	5
1.3	Matériel et logiciels à disposition.....	5
1.4	Prérequis	5
1.5	Cahier des charges	6
1.5.1	Objectifs et portée du projet (objectifs SMART)	6
1.5.2	Caractéristiques des utilisateurs et impacts	6
1.5.3	Fonctionnalités requises (du point de vue de l'utilisateur)	6
1.5.4	Contraintes.....	7
1.5.5	Travail à réaliser par la candidate	7
1.5.6	Méthodes de validation des solutions.....	8
1.6	Points évalués.....	8
2	Planification Initiale.....	9
3	Analyse	14
3.1	Opportunités	14
3.1.1	SWOT.....	14
3.2	Document d'analyse et conception	15
3.2.1	Diagramme de flux.....	15
3.2.2	Maquettes	18
3.2.3	Modèle logique de données.....	27
3.2.4	Dictionnaire des données	28
3.2.5	Description des entités et des relations.....	31
3.3	Conception des tests.....	33
3.4	Planification détaillée.....	36
3.4.1	Sprint 1	36
3.4.2	Sprint 2	38
3.4.3	Sprint 3	39
4	Réalisation	41
4.1	Dossier de Réalisation	41
4.1.1	Outils utilisés	41
4.1.2	Configuration de l'environnement.....	42
4.1.3	Modèle physique de données	43
4.1.4	Arborescence	44



4.2	Etapes de réalisations	46
4.2.1	Migrations & Models	46
4.2.2	Views Blades, Routes & Controllers.....	48
4.2.3	Request	49
4.2.4	Inscriptions – Connexion – Gestion de mot de passes.....	50
4.2.5	Graphique aléatoire de la page d'accueil principale	53
4.2.6	Barre de recherche	55
4.2.7	Graphiques combinaison.....	57
4.2.8	Importation de fichier CSV	60
4.2.9	CRUD & RUD	63
4.3	Modifications.....	64
5	Tests.....	65
5.1	Dossier des tests.....	65
5.2	Tests échoués, non réalisés & bugs.....	66
6	Conclusion	67
6.1	Bilan des fonctionnalités demandées.....	67
6.1.1	Suites à donner	68
6.2	Bilan de la planification.....	68
6.3	Bilan personnel	69
6.4	Remerciements	69
7	Divers	70
7.1	Journal de travail	70
7.2	Webographie	70
8	Annexes.....	72
8.1	Glossaire	72

1 Spécifications

1.1 Titre

DataVerse : Site web d'analyse et d'exploration de données météorologiques.

1.2 Description

Dans le cadre du travail pratique individuel (TPI), les candidats à l'obtention de leur CFC doivent réaliser un projet, commandé par leur chef de projet.

Ce projet, décrit dans ce présent rapport, a pour but la mise en œuvre d'un site web permettant l'affichage, le chargement et l'analyse de données météorologiques. Il est implémenté au travers du framework Laravel, avec des API supplémentaires si nécessaires.

1.3 Matériel et logiciels à disposition

1 Ordinateur standard (laboratoire de l'ETML)

1 Suite Microsoft Office

1 Environnement de développement web :

- VSCode
- Composer
- Framework Laravel
- Nodejs dans le cadre de l'utilisation du framework Tailwind
- Docker
- GitHub
- Mailtrap.io
- phpMyAdmin

1.4 Prérequis

Compétences en base de données → Modules 100, 104 et 105.

Compétences en développement Web → Modules 101, 120, 133 et 151.

Compétences en outils bureautiques → Module 302.

Compétences en gestion de projets → Modules 306 et 431.

1.5 Cahier des charges

1.5.1 Objectifs et portée du projet (objectifs SMART)

Au terme du projet, le lundi 3 juin 2024 à 15h45, la candidate rend, à ses experts et son chef de projet, un site web qui respecte les points techniques évalués spécifiques au projet, décrits au point 8 du cahier des charges. Elle rend également les livrables finaux tels que le présent rapport, le journal de travail et une archive complète. Le rendu se fait selon les modalités décidées dans le cahier des charges.

1.5.2 Caractéristiques des utilisateurs et impacts

Le profil d'utilisateur de ce site web peut être pour un chacun qui a un intérêt pour la météo et en particulier les recherches sur des données antérieures. Il ne s'agit donc pas d'un site prévisionnel de météo mais de recherches curieuses avec des possibilités de catégories, et de graphiques.

Etant donné que les utilisateurs peuvent être de tout âge, de n'importe quelle région géographique sans aucune distinction ; l'ergonomie du site permet une UI modeste, avec des couleurs sobres et une structure intuitive. Il n'a pas été recherché à d'avoir un design sophistiqué ou particulièrement recherché.

Par ailleurs, seul est employé le framework CSS Tailwind en combinaison de Laravel. Il n'y a pas de fonctionnalités avec du JavaScript comme on en trouve régulièrement dans les sites web modernes. Sauf dans le cas des graphiques, mais en l'occurrence, c'est l'API utilisée qui le déploie.

1.5.3 Fonctionnalités requises (du point de vue de l'utilisateur)

Il y a trois types d'utilisateurs décrits dans le cahier des charges, repris ci-dessous et modifié avec plusieurs précisions.

1. Les visiteurs du site peuvent :

- S'informer du but du site sur la page d'accueil.
- Visualiser les catégories de données.
 - Par exemple : précipitations, températures, ensoleillements, vents
- Visualiser les catégories géographiques.
 - Par exemple : continents, pays, régions, lieux
- Rechercher et visualiser des données en combinant les catégories. Il est possible de filtrer ces données selon une plage temporelle.
- S'inscrire sur le site au moyen d'un formulaire.
 - L'inscription suit une démarche sécurisée (par exemple : envoi d'un e-mail au visiteur avec un lien d'activation du compte valable 24h)
- Se loguer s'ils sont inscrits.
 - Le login est traité avec le sérieux sécuritaire actuel (par exemple : mdp chiffré, envoi d'un e-mail si mdp oublié)

2. Les contributeurs logués peuvent :
 - Gérer leur profil en modifiant des données personnelles, le mot de passe et en le désactivant.
 - Maintenir les données météorologiques dont ils sont propriétaires (CRUD).
 - Charger/remplacer un ensemble de données (dont ils sont propriétaires) par importation de fichier CSV.
3. Le ou les administrateurs du site peuvent :
 - Gérer les contributeurs (RUD).
 - Désactiver des données météorologiques erronées ou mettant en danger la sécurité et le sérieux du site.
 - Désactiver les contributeurs inactifs depuis plus de 5 ans.
 - Maintenir les données météorologiques de tous
 - Charger / remplacer les ensembles de données par importation de fichier CSV

1.5.4 Contraintes

La candidate est responsable de la sécurité de l'ensemble de son travail personnel individuel, qu'il s'agisse du rapport, du journal de travail, du code, etc.

En l'occurrence, le présent rapport est enregistré dans son OneDrive personnel, partagé avec le chef de projet et les experts, ainsi que d'autres documents comme les modèles conceptuels et logiques de la base de données, les images et captures d'écran. Le journal de travail est directement en lien avec le projet iceScrum, dont les différents interlocuteurs font également parties. De plus, tous les fichiers de code sont reliés à un repository GitHub publique dont le lien a été fourni à tous.

1.5.5 Travail à réaliser par la candidate

La candidate doit mettre en œuvre un site web sur le thème explicité dans les précédents chapitres. Cette mise en œuvre se fait à travers Laravel.

Le site devra contenir toutes les fonctionnalités demandées dans le cahier des charges. Si une fonctionnalité ne devait pas être implémentée, le rapport doit décrire pourquoi et comment on peut y remédier dans un futur hypothétique. L'aspect visuel, dit UI, du site respecte les caractéristiques du point 1.5.2.

Le rapport doit remplir les dix critères de l'évaluation des TPI au maximum et se doit être complet pour que le travail soit reproductible. L'apprenti doit également fournir un journal de travail et des planifications initiales et détaillées. Concernant la planification détaillée, celle-ci est disponible en tout temps pour les évaluateurs sur le projet iceScrum et sera intégrée au fur et à mesure dans le rapport.

1.5.6 Méthodes de validation des solutions

Dans le cadre de ce TPI, les tests effectués seront principalement des tests de types « uses cases » ou cas d'utilisation. Après chaque implémentation de fonctionnalité, des tests en lien avec les user stories décrites dans IceScrum, des tests de valeurs extrêmes, de valeurs limites seront effectués manuellement. L'ensemble de ces tests peut être considéré comme des tests fonctionnels. L'environnement sera l'environnement de développement de l'ETML.

Des tests unitaires et des tests d'intégration seraient également adaptés dans ce cadre, avec une automatisation ; tout comme des tests de charges lorsque le site est déployé. Cependant la candidate n'étant pas à l'aise avec ces procédés technologiques, il a été convenu avec le chef de projet que la méthodologie de tests resterait plus élémentaire.

1.6 Points évalués

En reprenant les points 7 et 8 du cahier des charges et en respectant les critères d'évaluation des TPI fournis, les points listés ci-dessous seront évalués :

- La planification initiale au terme du 1^{er} jour
- Le rapport
- Le journal de travail
- Une archive, contenant :
 - Le site complet (données sources, code source, sql, etc)
 - Un guide de mise en service du site de max 1page A4 r/v
 - Un guide d'utilisation de mise en service du site de max 1page A4 r/v
- La présentation au terme du TPI

Les points techniques spécifiques évalués sont :

- Le respect des exigences visuelles (UI)
- Le MLD définitif de la base de données
- La fonctionnalité de recherche des données météo
- La fonctionnalité d'inscription des contributeurs
- La fonctionnalité de maintenance des données météo des contributeurs (CRUD)
- La fonctionnalité de chargement/remplacement des données météo par fichier CSV.
- La fonctionnalité de gestion des contributeurs (RUD)

2 Planification Initiale

Pour la planification en générale, la méthodologie AGILE au travers du site de iceScrum a été mise en place. Les différentes parties prenantes de ce projet ont été invitées au projet détenu par la candidate. Ils peuvent donc en tout temps visualiser les différents sprints et les user stories ainsi que les tâches qui y sont inscrites. Les sprints débutent et se terminent lors des rendez-vous hebdomadaires entre la candidate et le chef de projet.

Le projet de TPI a une durée de 3 semaines environ, du 13 mai 2024 au 03 juin 2024, pour une durée de 88 heures. Le lundi 20 mai 2024 est hors planification car il s'agit d'un jour férié.

Plusieurs « features » ont été déterminées lors de la planification initiale. Ces features reprennent les thèmes généraux du projet complet et se répartissent tout au long de la planification les différentes user stories et les tâches associées. Les features sont inscrites en gras dans les tableaux décrivant les sprints.

Le Sprint 1 débute le 13 mai et se termine le 22 mai au matin, le Sprint 2 est du 22 mai au 27 mai et enfin le 3^{ème} Sprint se déroule du 27 mai au 03 juin.

En analysant la planification (tableau récapitulatif ci-dessous), on peut constater qu'il y a un dépassement d'environ 1h sur le total des heures allouées pour le TPI. Cependant, les heures de rendez-vous avec les experts et les rencontres hebdomadaires avec le chef de projet ont été comptabilisées dans la partie documentation, ce qui peut expliquer ce dépassement global. De plus plusieurs parties dépassent leur pourcentage initial, mais il est important de noter que parfois la partie Analyse et Documentation peuvent se confondre. En effet lors d'une recherche sur un élément, il sera annoté dans le rapport dans la foulée. De plus les parties Implémentation et Tests peuvent également se mélanger au vu de la méthode de validation choisie dans le point concernée.

De plus, l'utilisation de l'outil IceScrum ne permet pas d'allouer des heures concrètement, donc cette planification a été fait à travers une estimation globale des features sans tenir compte d'une planification plus détaillée. Finalement, il est important de noter que bien que les critères d'évaluation des TPI demandent des estimations entre 2 et 4heures, la fourchette utilisée ici varie de 1 à 5h pour des raisons d'essais de conformité à la réalité ainsi qu'au respect des heures allouées par jour.



Tableau récapitulatif avec les heures sur les 3 Sprint

Planning CDC	Features	Total heures planifiées	Total heures accordées par le CDC	Différence en heures
Documentations 25%	Documentations	26h20 (26.33)	22	+4h20
Analyses 15%	Maquettes	8h20 (8.33)	13,2	-4h40 (4,87)
	Base de données			
	RFC – bonnes pratiques			
Implémentations 45%	Base de données	40h	39,6	+0h25 (0,4)
	Inscriptions – Mots de passes – Connexion			
	Gestion CRUD – Contributeurs			
	Gestion RUD – Admin			
	Recherches & Graphiques			
	Importations CSV			
	Déploiement			
Tests 15%	Toutes les implémentations	14h30 (14,5)	13,2	+1h20 (1,3)
		89h10	88h	+1h10

Tableau 1 Récapitulatif des heures de planification



Sprint 1 13.05.2024 – 22.05.2024						
Lundi 13	Mardi 14	Mercredi 15	Jeudi 16	Vendredi 17	Mardi 21	Mercredi 22
Démarrage TPI 1h Documentations - Préparation documents, GitHub, iceScrum, JDTV - Planification initiale 5h20	Documentations - Finalisation planification initiale 1h Analyses - Maquettes 5h Documentations - Rédaction du rapport - JDTV 1h	Analyses : Base de données - MCD – MLD 2h20	Implémentations : Base de données - Préparer fichiers migrations et routes associées 4h Inscriptions – Mots de passes – Connexions - View 1 formulaire inscription et des routes 2h Tests : Inscriptions - Tests dans la base de données 1h	Implémentations : Inscriptions – Mots de passes – Connexions - Envoi email de validation 2h - Gestion des mots de passes 4h	Implémentations : Inscriptions – Mots de passes – Connexions - Inscriptions admin et contributeurs, modification mot de passes et oublis 3h Tests : inscriptions - Tests sur toutes les valeurs limites 2h Documentations - Rédaction du rapport - JDTV 1h	RDV CdP : Sprint Retrospective – Démarrage Sprint 2 1h Documentations - Rédaction du rapport 1h Analyses - Recherche sur les différentes RFC et bonnes pratiques pour les données 1h



Sprint 2 22.05.2024 – 27.05.2024			
Mercredi 22	Jeudi 23	Vendredi 24	Lundi 27
RDV CdP : Sprint Retrospective – Démarrage Sprint 2 1h <u>Documentations</u> - Rédaction du rapport 1h <u>Analyses</u> - Recherche sur les différentes RFC et bonnes pratiques pour les données 1h	RDV Expert 2 8h30 1h <u>Implémentations :</u> Recherches & Graphiques - View principale – routes – controller pour la recherche 4h Importations CSV - View – route – controller pour l'import 2h30	<u>Implémentations :</u> Gestion CRUD – Contributeurs - Views – routes – controller pour le CRUD données personnelles sauf mot de passes 3h <u>Tests : Importations CSV</u> - Tests sur l'import et la manipulation des données par le contributeur 4h30	RDV CdP : Sprint Retrospective – Démarrage Sprint 3 1h <u>Implémentations :</u> Gestion CRUD – Contributeurs - Views – routes- controller pour le mot de passe 2h Gestion RUD – Admin - Views – routes – controller pour la gestion des données personnelles, mdpasse y compris. 2h <u>Tests : Gestion RUD</u> 1h



Sprint 3 27.05.2024 – 03.06.2024					
Lundi 27	Mardi 28	Mercredi 29	Jeudi 30	Vendredi 31	Lundi 03
RDV CdP : Sprint Retrospective – Démarrage Sprint 3 1h <u>Implémentations :</u> Gestion CRUD – Contributeurs - Views – routes- contrôler pour le mot de passe 2h Gestion RUD – Admin - Views – routes – contrôler pour la gestion des données personnelles, mdpasse y compris. 2h <u>Tests : Gestion RUD</u> 1h	<u>Implémentations :</u> Gestion CRUD – Contributeurs - Finalisation des différents éléments 2h Gestion RUD – Admin - Finalisation des différents éléments 1h <u>Tests globaux</u> - Finalisation des différents tests pas encore effectués ou résolus 3h	<u>Documentations</u> - Rédaction du rapport et JDTV 3h	Implémentations : Déploiement - Essais de déploiement du site sur meteo.section-inf.ch 4h <u>Documentations</u> - Rédaction du guide de mise en service 1h - Rédaction du rapport et JDTV 1h	<u>Documentations</u> - Rédaction du guide d'utilisation du site 2h - Rédaction finale du rapport et JDTV 4h	RDV CdP : Sprint Retrospective Finale – 1h <u>Tests sur l'intégralité du site et vérification du code</u> 3h <u>Documentations</u> - Révision finale de tous les livrables. 1h Rendus aux parties concernées selon modalités 1h

Tableau 2 Planification initiale

3 Analyse

3.1 Opportunités

3.1.1 SWOT

L'analyse SWOT ou FFMO en français est un outil qui permet de déterminer les forces, les faiblesses, les opportunités et les menaces en lien avec un projet et ainsi d'identifier les facteurs internes et/ou externes à la réalisation du dit projet. Il est beaucoup utilisé dans les stratégies d'entreprises.

Il apparaît judicieux de pouvoir effectuer un SWOT dans le cadre de ce projet de TPI afin de déterminer toutes les difficultés potentielles mais également les possibilités de solutions et d'évolutions.

Facteurs internes	
STRENGTHS	WEAKNESSES
<ul style="list-style-type: none"> • Capacité d'intégration rapide • Capacité de créativité • Connaissances du framework Laravel • Projet qui renforce les connaissances du développement web et de base de données • Aisance rédactionnelle et analytique 	<ul style="list-style-type: none"> • Remise en en situation de travail réel après 3 ans • Difficulté dans la gestion du temps • Priorisation parfois imprécise <ul style="list-style-type: none"> ◦ Nécessité d'avoir des tâches découpées ◦ Planification initiale complexe à projeter
Facteurs externes	
OPPORTUNITIES	THREATS
<ul style="list-style-type: none"> • Pas de site web de ce style sur le marché, gratuit et simple • Partage d'informations sans aucune contrainte politique, économique, géographique • Utilisateurs de toute provenance 	<ul style="list-style-type: none"> • Hébergement du site : <ul style="list-style-type: none"> ◦ Beaucoup de données <ul style="list-style-type: none"> ▪ Coûts ▪ Déploiement complexe ▪ Maintenance ▪ Charge visiteurs • Sécurité des données personnelles avec possibles failles • Visiteurs malicieux qui volontairement fournissent des données erronées <ul style="list-style-type: none"> ◦ Gestion de la base de données qui demandent des compétences techniques

3.2 Document d'analyse et conception

3.2.1 Diagramme de flux

Pour représenter de manière schématique le fonctionnement du site et de la logique de programmation dans le backend du site, un diagramme de flux a été créé avec le site draw.io.

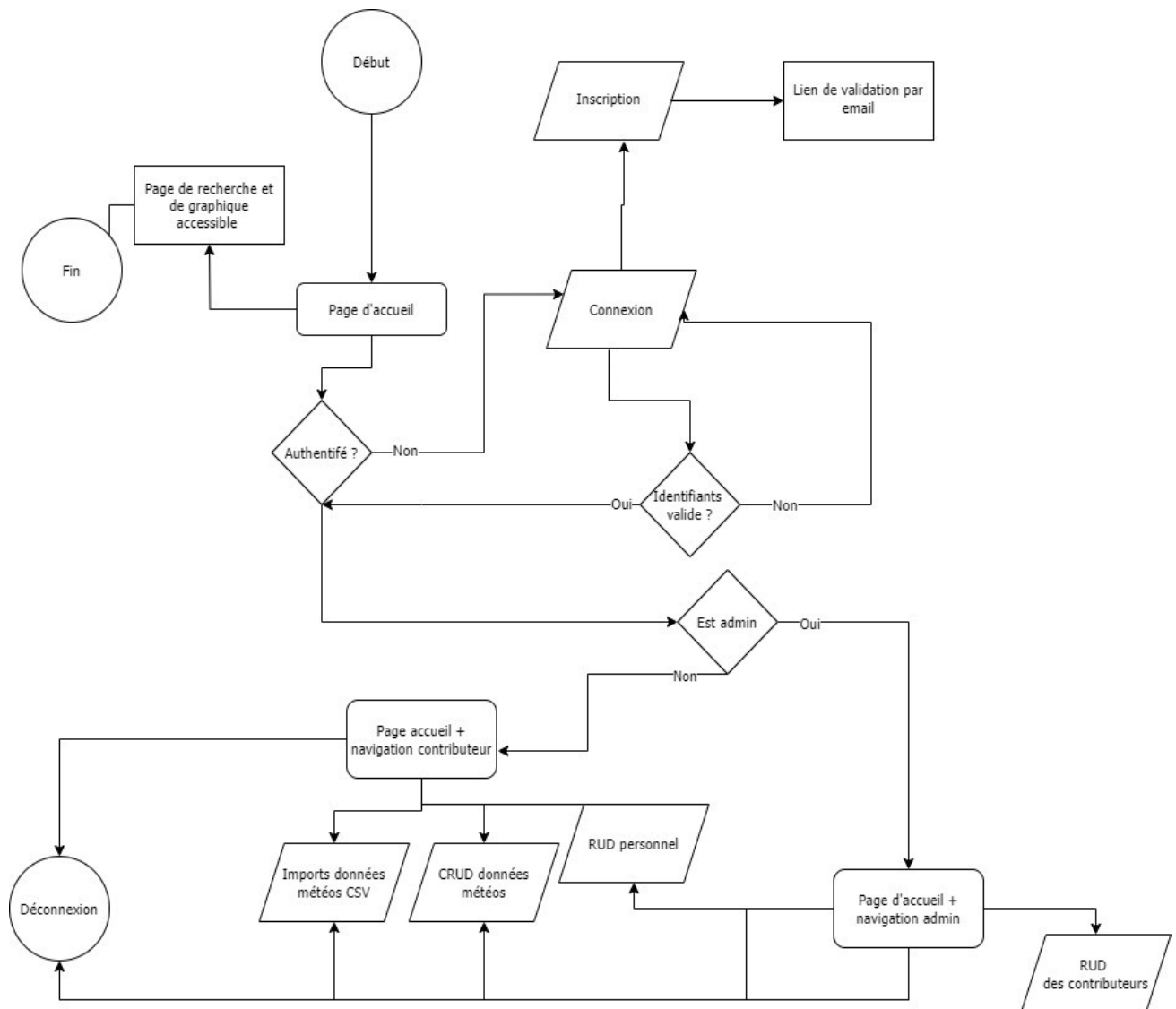


Figure 1 Diagramme de flux

Ce diagramme présente les différentes possibilités de navigation et par là même, les différents processus qui agissent en arrière-plan. Il est détaillé ci-dessous, en plusieurs parties distinctes qui permettent de distinguer les divers composants.

Le diagramme commence lorsqu'un utilisateur va aller sur le site par l'URL de son navigateur. Il se termine soit par un arrêt de la navigation, soit par une déconnexion de l'utilisateur authentifié.

En haut à gauche, le visiteur anonyme, qui a un accès standard avec une possibilité d'effectuer des recherches, d'avoir accès à des graphiques ; sa navigation peut s'arrêter là.

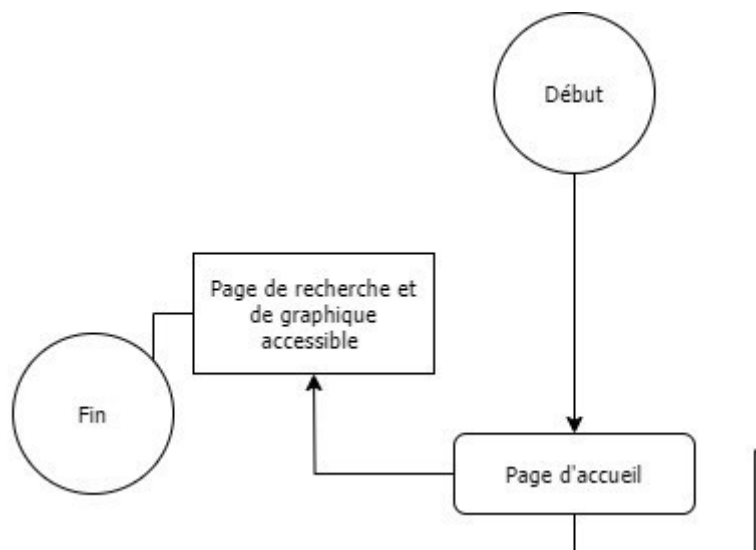


Figure 2 Utilisateur anonyme - Diag. Flux

Sur la page d'accueil, il va y avoir une vérification de l'utilisateur, s'il est authentifié ou non. S'il ne l'est pas il aura une possibilité de se connecter ou de s'inscrire. De là une validation des identifiants est effectuée. Si les identifiants dans la base de données ne sont pas reconnus ou n'existent pas, l'utilisateur est renvoyé à la page de connexion avec un message d'erreur approprié.

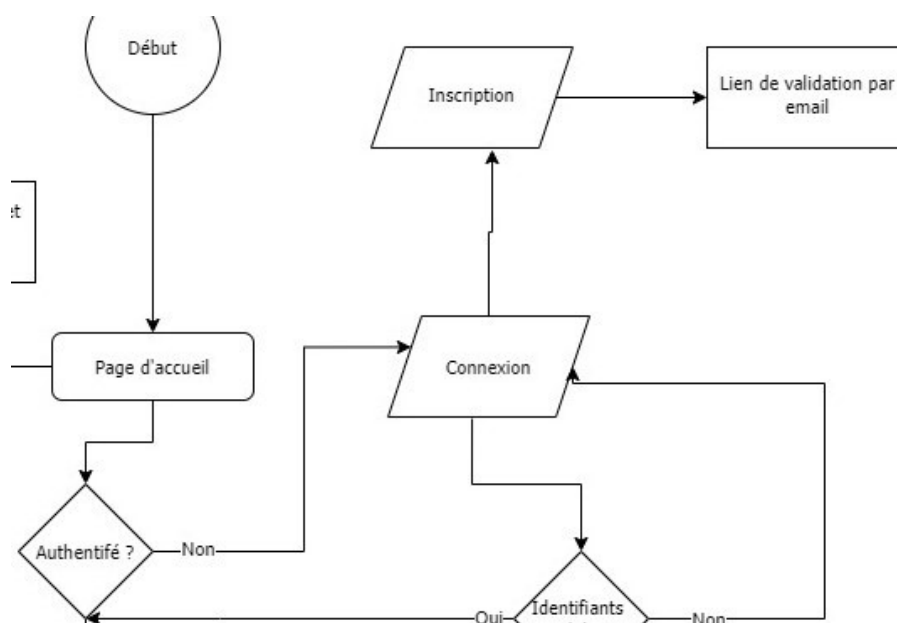


Figure 3 Authentification - Diag. Flux

Si l'utilisateur a des identifiants valides, il est donc authentifié sur le site avec un « jeton » de session. Là une deuxième interrogation se pose en base de données : l'utilisateur a-t-il un statut d'administrateur ou non ? Si ce n'est pas le cas, il a accès au site avec la navigation « contributeur » et des accès restreints, bien que plus importants que l'utilisateur anonyme.

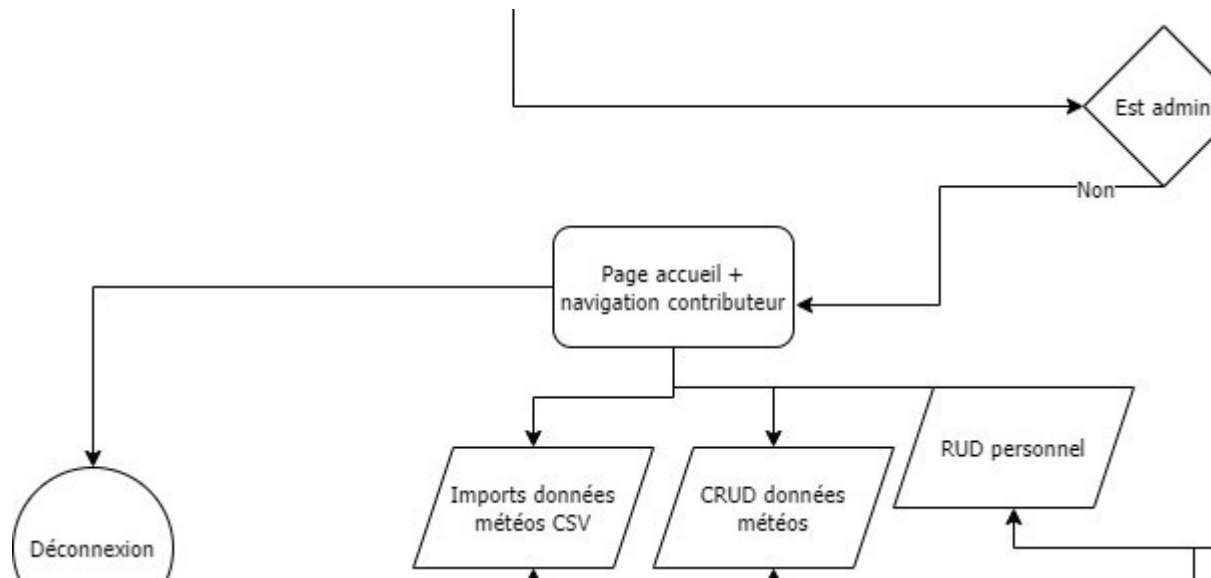


Figure 4 Contributeur - Diag. Flux

Enfin, si l'utilisateur authentifié est un administrateur, il accède à la navigation dite « admin » et a une fonctionnalité supplémentaire du contributeur, la gestion RUD des contributeurs.

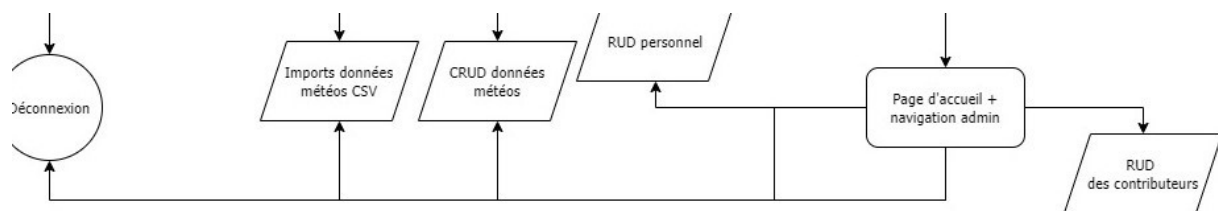


Figure 5 Administrateur - Diag. Flux

3.2.2 Maquettes

Afin de conceptualiser l'identité visuelle, de présenter les fonctionnalités et les interactions entre les différents contenus du site web, plusieurs maquettes ont été créées avec l'application de Figma. Ces maquettes serviront également de modèles à la conception frontend des views dans Laravel.

Chaque view du site se compose de deux éléments récurrents. Un « header » avec un logo, une barre de navigation qui se modifie en fonction de l'authentification de l'utilisateur et par conséquent un bouton de connexion ou de déconnexion. Le bouton Accueil permet de revenir à cette page, le logo également. Le « footer » lui ne change pas et contient les crédits du site. Le body est le seul élément qui se modifie au travers de la navigation sur le site. Ces différents éléments sont décrits ci-dessous avec illustrations.

La page d'accueil principale contient une barre de recherche de lieu. Une fois le lieu trouvé (ou non si pas existant dans la base de données), l'utilisateur est amené vers une view correspondante.

Un message de bienvenue et d'explications sur le site ainsi qu'un graphique généré aléatoirement composent le reste du contenu de cette view.

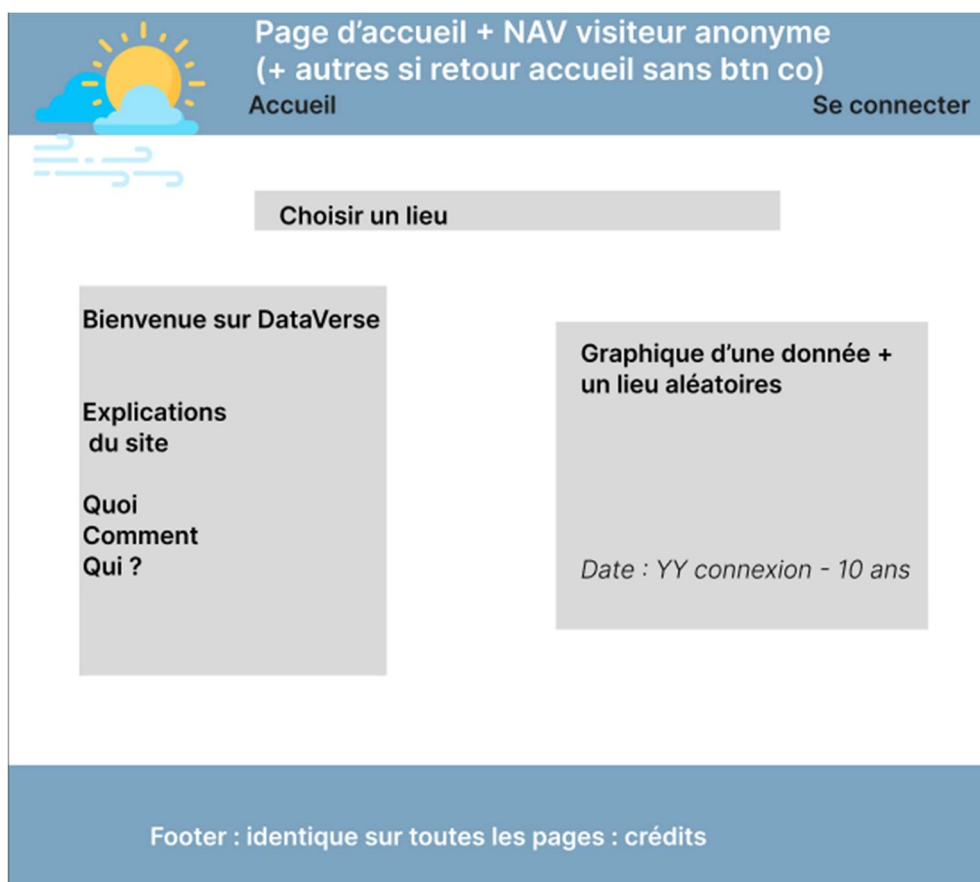



Figure 6 Accueil principal – Maquettes

Quand l'utilisateur va choisir un lieu dans la barre de recherche, il est redirigé sur la page des graphiques de ce lieu. Elle contient toujours la barre s'il veut changer de lieu. L'utilisateur peut choisir le type de données météorologiques qui l'intéresse et la ligne temporelle qui l'intéresse.



Combinaison graphiques (tous les users)

Accueil Se connecter | Prénom + Nom + btn déconnexion

Nom du lieu choisi

Choisir un lieu

Catégories météo à cocher

☐ ☐ ☐ ☐

Graphique(s) généré(s) par les catégories choisies

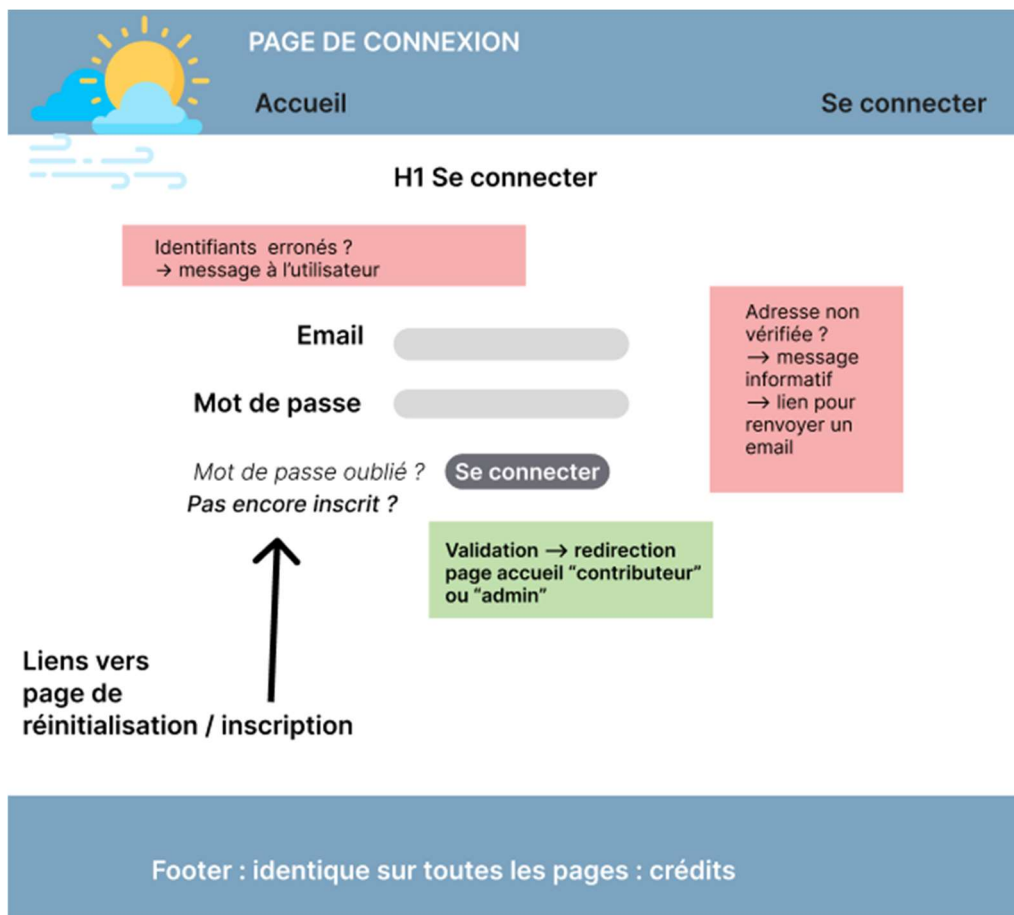
Ligne temporelle à choisir

Footer : identique sur toutes les pages : crédits

Figure 7 Graphiques - Maquette

Si l'utilisateur veut se connecter, il est dirigé sur cette view de connexion. Elle se compose d'un formulaire simple. Lorsqu'il va entrer ses identifiants, un processus de vérifications multiples va s'effectuer au niveau de la base de données et amener ainsi plusieurs actions.

- 1) Les identifiants sont corrects
 - a. L'utilisateur est redirigé sur sa page d'accueil personnalisée selon s'il est contributeur ou administrateur.
- 2) Les identifiants sont incorrects
 - a. L'adresse ou le mot de passe sont erronés → message d'erreur en conséquence
 - b. L'adresse email n'a pas été validée → message d'erreur et proposition de renvoyer un lien
- 3) Les identifiants ne sont pas reconnus
 - a. L'adresse email n'existe pas dans la base de données → message d'erreur en conséquence



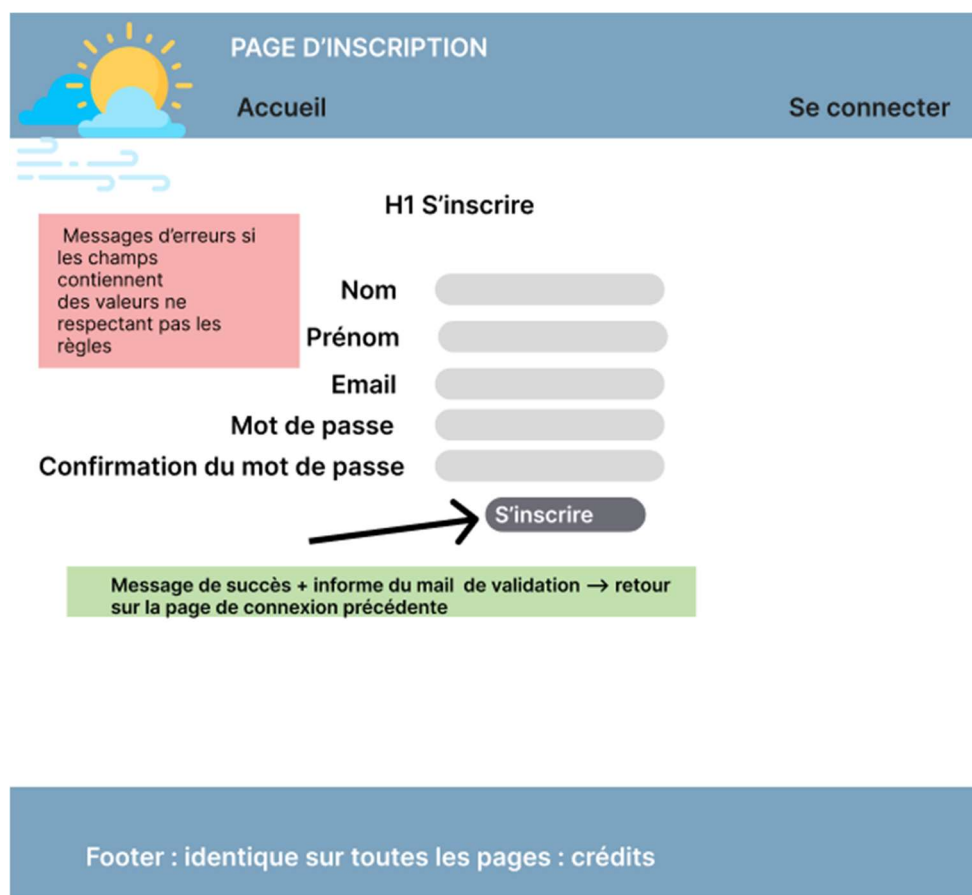
La maquette de la page de connexion est structurée comme suit :

- En-tête (Barre bleue) :**
 - À gauche : Logo météo (soleil, nuages, vent).
 - Centre : **PAGE DE CONNEXION**
 - Droite : **Accueil** (liens) et **Se connecter** (bouton).
- Contenu principal :**
 - Titre : **H1 Se connecter**
 - Message d'erreur (rouge) : **Identifiants erronés ?** → message à l'utilisateur
 - Champs de saisie : **Email** et **Mot de passe** (champs gris).
 - Message d'erreur (rouge) : **Adresse non vérifiée ?** → message informatif → lien pour renvoyer un email
 - Textes : *Mot de passe oublié ?* et *Pas encore inscrit ?*
 - Bouton : **Se connecter** (gris foncé)
 - Message de validation (vert) : **Validation → redirection page accueil "contributeur" ou "admin"**
 - Texte avec flèche : **Liens vers page de réinitialisation / inscription**
- Pied de page (Barre bleue) :** **Footer : identique sur toutes les pages : crédits**

Figure 8 Connexion – Maquette

Si l'utilisateur veut s'inscrire, il est dirigé via la page de connexion sur une page d'inscription. Elle contient un formulaire et un bouton d'inscription.

Si les champs contiennent des valeurs qui ne respectent pas les règles, l'utilisateur doit les corriger selon le message d'erreur correspondant qui se sera affiché. Les champs seront alors préremplis des anciennes valeurs qu'il avait inscrit sauf le mot de passe.



La maquette de la page d'inscription est présentée sur un fond bleu clair. En haut à gauche, il y a un logo d'un soleil derrière des nuages. À droite, le titre "PAGE D'INSCRIPTION" est écrit en blanc. En dessous, les liens "Accueil" et "Se connecter" sont également en blanc. Le titre principal de la section est "H1 S'inscrire". À gauche du formulaire, un message d'erreur est affiché dans une boîte rose : "Messages d'erreurs si les champs contiennent des valeurs ne respectant pas les règles". Le formulaire lui-même est composé de cinq champs : "Nom", "Prénom", "Email", "Mot de passe" et "Confirmation du mot de passe", chacun avec un champ de saisie gris. À droite du dernier champ, il y a un bouton "S'inscrire" en gris foncé. Une flèche noire pointe vers ce bouton. En dessous du bouton, un message de succès est affiché dans une boîte verte : "Message de succès + informe du mail de validation → retour sur la page de connexion précédente". En bas de la page, un footer bleu clair contient le texte "Footer : identique sur toutes les pages : crédits".

Figure 9 Inscription - Maquettes

Une fois l'utilisateur connecté, il est automatiquement dirigé sur sa page d'accueil personnelle.

Celle-ci se compose d'une nouvelle barre de navigation dans le header qui affiche en sus son prénom et son nom, un bouton de déconnexion, un bouton de gestion de profil personnel, s'il est un contributeur.

Le contenu de la view est assez simple. Il contient la même barre de recherche que la page d'accueil pour les lieux. Il contient un bouton de gestion des données météorologiques que le contributeur a pu importer ainsi qu'une liste-liens de la référence des cinq dernières données.

Enfin, la view comprend le formulaire d'importation de fichiers. Si le fichier est correctement importé, un message de succès apparaît sur la view et la liste-liens se modifie en conséquence. Dans le cas contraire, un message d'erreur explicite est transmis (par exemple, mauvais format de fichier).

Page accueil + NAV "Contributeur"

Prénom + Nom
Déconnexion

Accueil Gérer mon profil

Message : Vous êtes bien connecté si 1ère redirection

Choisir un lieu

Gérer mes données

Liste "liens" avec les données du contributeur

Importer des données

Fichier

Importer

Footer : identique sur toutes les pages : crédits

Figure 10 Accueil Contributeur - Maquette

La page d'accueil pour l'administrateur est assez similaire à celle du contributeur. Elle contient deux points supplémentaires ; la gestion des contributeurs via une barre de recherche et un lien vers la base de données pour la gestion des données corrompues ou obsolètes. L'idée est que l'administrateur gère directement ces données depuis la base de données.

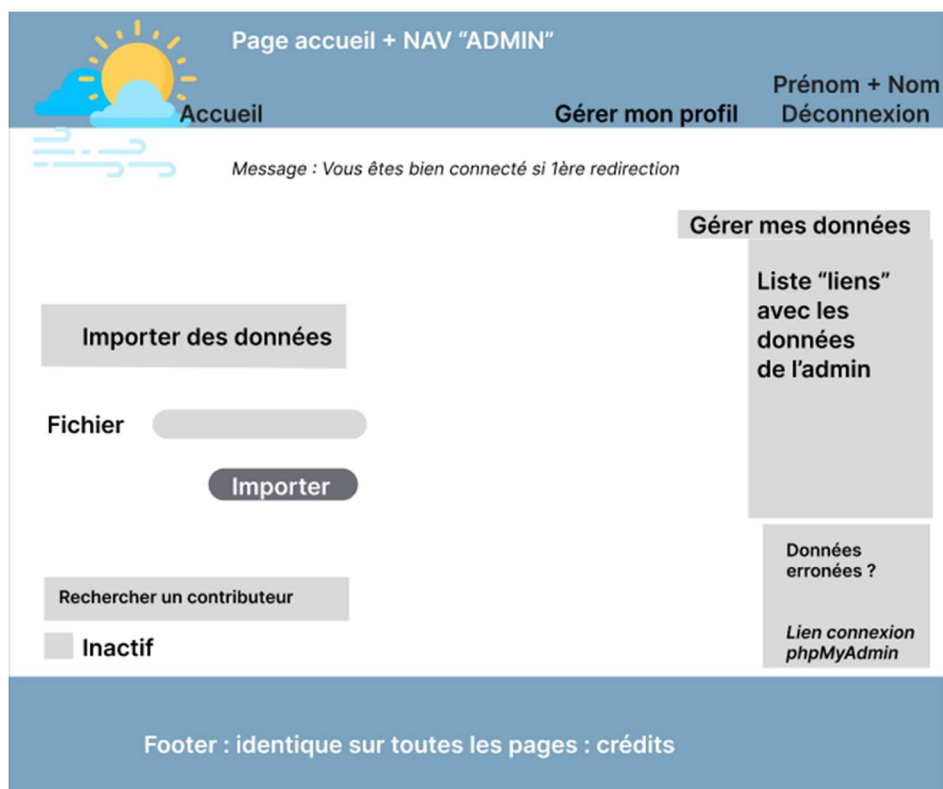



Figure 11 Accueil – Administrateur – Maquette

Les utilisateurs authentifiés peuvent gérer les données météorologiques qu'ils ont eux-mêmes importés au travers « Gérer mes données » ou de la liste-liens. Gérer mes données est une liste cachée de toutes les données de l'utilisateur. Après avoir sélectionné celle qu'il veut traiter, il est redirigé sur la view qui y est attachée.

Il retrouve le nom du fichier, la date d'importation, une possibilité de recharger un fichier pour ces données ou de les supprimer. Il a également un aperçu des graphiques qui seront visibles pour tous les utilisateurs.



Gestion CRUD "Contributeur & Admin"

Prénom + Nom

AccueilGérer mon profilDéconnexion

H1 Données X

Nom du fichier

Date d'importation

Recharger un fichier

Fichier

Importer

Catégories météo à cocher

☐☐☐☐

Graphique généré par les catégories choisies

Supprimer

Ligne temporelle à choisir

Footer : identique sur toutes les pages : crédits

Figure 12 CRUD - Maquette

Les utilisateurs authentifiés peuvent gérer leurs données personnelles au travers de Gérer mon profil. Ils peuvent modifier leur profil, leur mot de passe ou supprimer leur profil. Ces fonctionnalités s'effectuent au travers de formulaires.

Figure 13 CRUD Personnel - Maquette

La suppression est en réalité, au sein de la base de données, une désactivation. En effet, dans la réalité, les données sont rarement supprimées définitivement pour des questions d'archivages et de sécurité. De plus, si l'utilisateur s'est trompé, il peut contacter l'administrateur et demander à rétablir son profil.

L'administrateur peut, en plus du reste, gérer les contributeurs. C'est-à-dire qu'il a un accès à leur profil personnel mais sans pouvoir effectuer de modifications des données. Il ne peut que réactiver le profil si besoin, ou de le désactiver, à la demande de l'utilisateur ou s'il est inactif depuis cinq ans. Il a également la possibilité de lui envoyer un lien de réinitialisation de mot de passe en cas de perte de ce dernier.

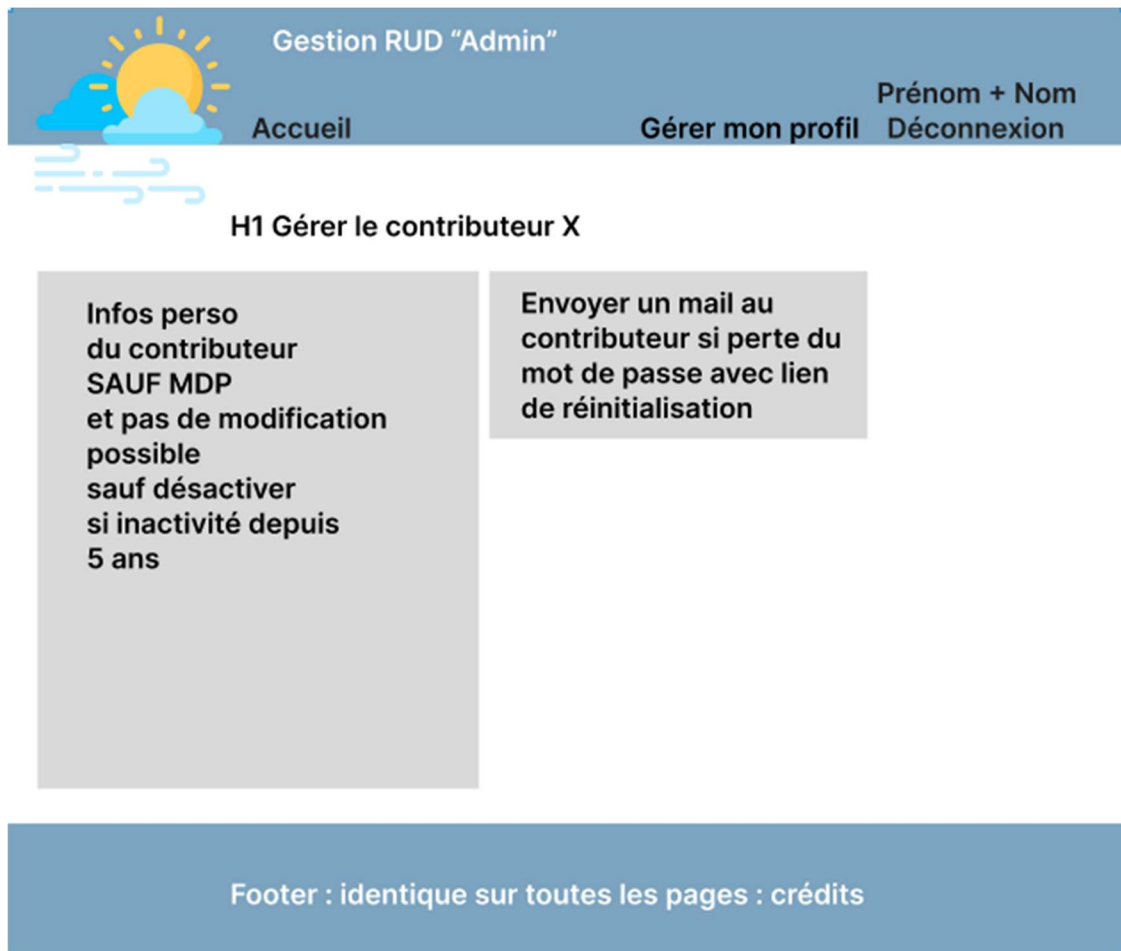
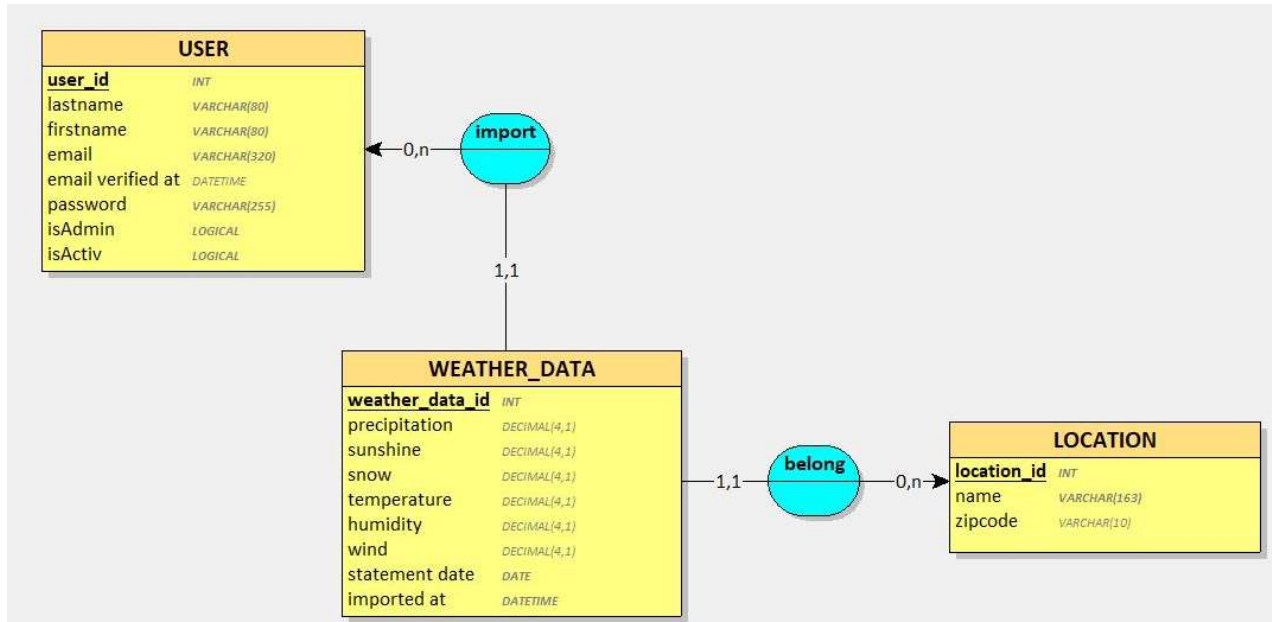


Figure 14 RUD Administrateur - Maquette

3.2.3 Modèle logique de données

Quand il y a une base de données, il y a forcément un modèle conceptuel (MCD) des données duquel découle le modèle logique de données (MLD).

Modèle conceptuel de données



Modèle logique de données

Le modèle logique respecte les trois formes normales. La première forme normale indique que chaque entité possède un identifiant. De plus, chaque propriété ne peut prendre qu'une seule valeur et doit être élémentaire. Les tables respectent donc cette première forme, car elles contiennent toute un identifiant unique et les propriétés sont atomiques.

La deuxième forme normale émet que les propriétés d'une entité doivent dépendre de tout l'identifiant. C'est-à-dire que dans les modèles de ce projet, les tables ne contiennent qu'un seul identifiant unique. Par exemple, dans le MLD, les propriétés de la table « weather_data » ne dépendent que de son identifiant propre et non pas de la localisation, qui est une table à part entière. La deuxième forme normale permet d'éviter ainsi les principales sources de redondances dans les relations.

Enfin la troisième forme normale notifie que les propriétés ne doivent pas dépendre d'une autre propriété que celle retenue comme identifiant. Pour mettre cela en place, il est nécessaire de travailler les dépendances entre les tables et d'ajouter des clés étrangères. Dans ce projet, la table « weather_data » dépend de deux autres tables et donc possède les deux clés étrangères de ces tables. Le modèle logique respecte ainsi la troisième forme normale.

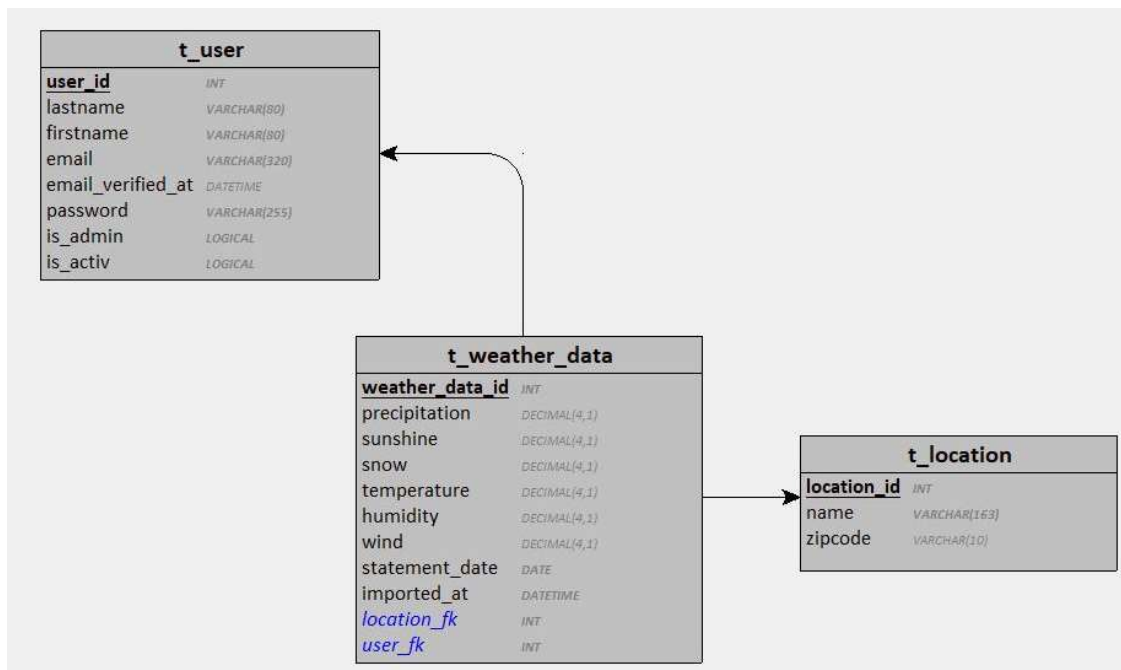


Figure 15 Modèle logique des données complet

3.2.4 Dictionnaire des données

Les entités du modèle logique de données contiennent chacune plusieurs propriétés aux types différents. Des tableaux descriptifs des propriétés pour chaque table sont ci-dessous. Pour les types choisis et leur longueur, plusieurs recherches ont été effectuées afin d'être au plus juste possible des normes RFC et des bonnes pratiques.

Données pour les noms, prénoms, labels

Il n'existe pas, à notre connaissance, de RFC concernant les longueurs pour les noms et les prénoms. De manière générale, la taille courante de ces propriétés est entre 50 et 100 VARCHAR. Il a été choisi 80 car il permet d'avoir pratiquement toutes les possibilités sans que cela prenne trop de mémoire.

Longueur maximale pour les adresses électroniques

Selon la RFC 5321, la longueur totale d'une adresse de courriel doit être de 64 octets pour la partie locale et de 255 pour la partie du domaine, ce qui fait un total de 319 caractères arrondis à 320.

Données pour les données météorologiques

Les données météorologiques sont régulièrement dans des formats et des normes particuliers établis, utilisés par les météorologues, les océanographes et les scientifiques de cette branche ; tels que NetCDF ou GRIB.

Cependant pour des raisons de totale lacune de ces formats, de simplicité, et de bon sens finalement, le type choisi pour ces données météorologiques est les DECIMAL.

Il est d'une longueur de 4,1. C'est-à-dire que 4 chiffres dont 1 après la virgule. Que cela soit des millimètres pour les précipitations ou des pourcentages pour le vent, il y a peu de chances que les valeurs de ces données soient supérieures à 3 chiffres avant la virgule.

Dates et l'heure

La RFC 3339 décrit comment les dates et les heures peuvent être formatées sur Internet et garder une cohérence entre tous. Elle est basée sur la norme ISO 8601 qui indique comment ces données doivent être représentées pour être au plus juste.

Le site de Microsoft indique quelle est la longueur du champ pour un type date-time, qui est de 24 et pour le type timestamp c'est 17¹.

Nom du lieu et codes postaux

Afin d'être certain de pouvoir prendre en compte tous les noms de lieux qui pourrait être inscrit, des recherches ont été faites sur les noms de lieux les plus longs du monde. Il se trouve que le plus grand compte 163 caractères.

Concernant les codes postaux, ils peuvent différer de continents à l'autre, contenir des lettres combinées à des chiffres et des traits d'union. C'est pourquoi il a été décidé de convenir du type VARCHAR et d'une longueur de 10 qui devraient normalement couvrir toutes les possibilités.

Tableaux des entités

T_user				
Nom de la propriété	Type	Longueur du champ	Contraintes	Description
user_id	integer	10	Auto-increment, NOT NULL, Unique, primary key	Clé primaire de l'utilisateur
name	varchar	80	NOT NULL	Nom de famille de l'utilisateur
firstname	varchar	80	NOT NULL	Prénom de l'utilisateur
email	varchar	320	NOT NULL	Email de l'utilisateur
email_verified_at	date-time	24		Date et heure de la vérification de l'email
password	varchar	255	NOT NULL	Mot de passe de l'utilisateur qui est haché dans la base
is_admin	boolean	1	NOT NULL	Vrai si l'utilisateur est administrateur
is_activ	boolean	1	NOT NULL	Vrai si l'utilisateur est actif

Tableau 3 T_USER - DICTIONNAIRE

¹ Dans le modèle logique de données, dans la table t_weather_data, la propriété imported_at est notifiée date-time car le logiciel looping qui a été utilisé pour créer les MCD MLD n'offre pas la possibilité d'utiliser timestamp. En l'occurrence c'est un timestamp qui a été choisi dans les migrations de Laravel.

T_weather_data				
Nom de la propriété	Type	Longueur du champ	Contraintes	Description
<i>weather__data_id</i>	integer	10	Auto-increment, NOT NULL, Unique, primary key	Clé primaire
<i>precipitation</i>	decimal	4,1		Valeurs des précipitations, en millimètres
<i>sunshine</i>	decimal	4,1		Valeurs d'ensoleillement, en minutes
<i>snow</i>	decimal	4,1		Valeurs d'enneigement, en centimètres
<i>temperature</i>	decimal	4,1		Valeurs de température extérieure, en degré Celsius
<i>humidity</i>	decimal	4,1		Valeurs du taux d'humidité, en pourcentage
<i>wind</i>	decimal	4,1		Valeurs du vent, en km/h
<i>statement_data</i>	date	24	NOT NULL	Date et heure du relevé des données
<i>imported_at</i>	timestamp	17	NOT NULL	Date et heure de l'importation des valeurs
<i>locatifon_fk</i>	int	10	NOT NULL	Clé étrangère qui relie la valeur au lieu
<i>user_fk</i>	int	10	NOT NULL	Clé étrangère qui relie la valeur à l'utilisateur.

Tableau 4 T_WEATHER_DATA – DICTIONNAIRE

T_location				
Nom de la propriété	Type	Longueur du champ	Contraintes	Description
<i>location_id</i>	integer	10	Auto-increment, NOT NULL, Unique, primary key	Clé primaire du lieu
<i>name</i>	varchar	163	NOT NULL	Nom du lieu
<i>zipcode</i>	varchar	10		Code postal du lieu

Tableau 5 T_LOCATION - Dictionnaire

3.2.5 Description des entités et des relations

Table USER et table WEATHER_DATA

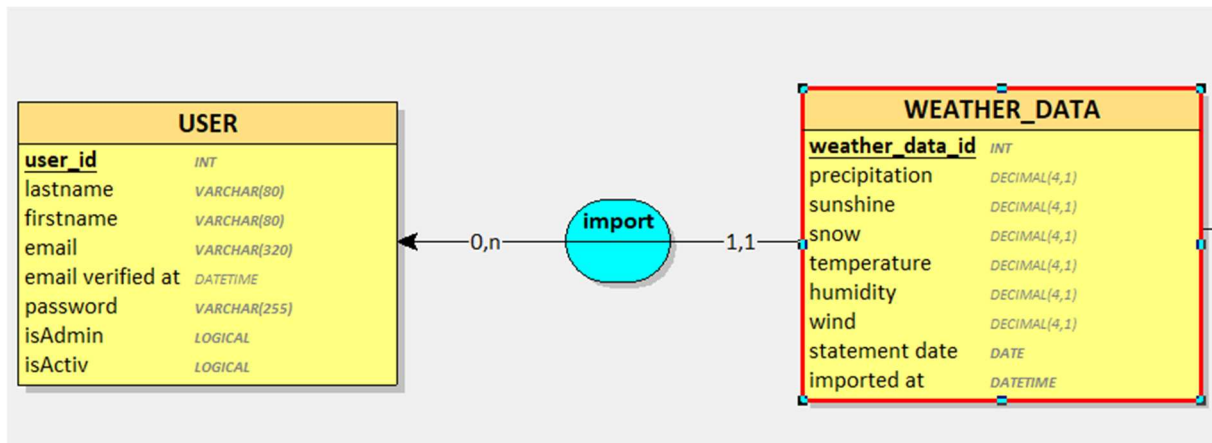


Figure 16 Relation USER to WEATHER_DATA

Cardinalité :

- Un utilisateur peut importer 0 à un nombre N de données météorologiques
- Un set de donnée météorologiques est importé par un seul utilisateur

Relation :

La relation exprime le lien d'import entre l'utilisateur et les données météorologiques.

MLD :

La table t_weather_data prend donc en clé étrangère l'identifiant de l'utilisateur, illustré ci-dessous en rouge.

t_weather_data	
<u>weather_data_id</u>	INT
precipitation	DECIMAL(4,1)
sunshine	DECIMAL(4,1)
snow	DECIMAL(4,1)
temperature	DECIMAL(4,1)
humidity	DECIMAL(4,1)
wind	DECIMAL(4,1)
statement_date	DATE
imported_at	DATETIME
location_fk	INT
user_fk	INT

Figure 17 USER_FK et LOCATION_FK dans t_weather_data

Table LOCATION et table WEATHER_DATA

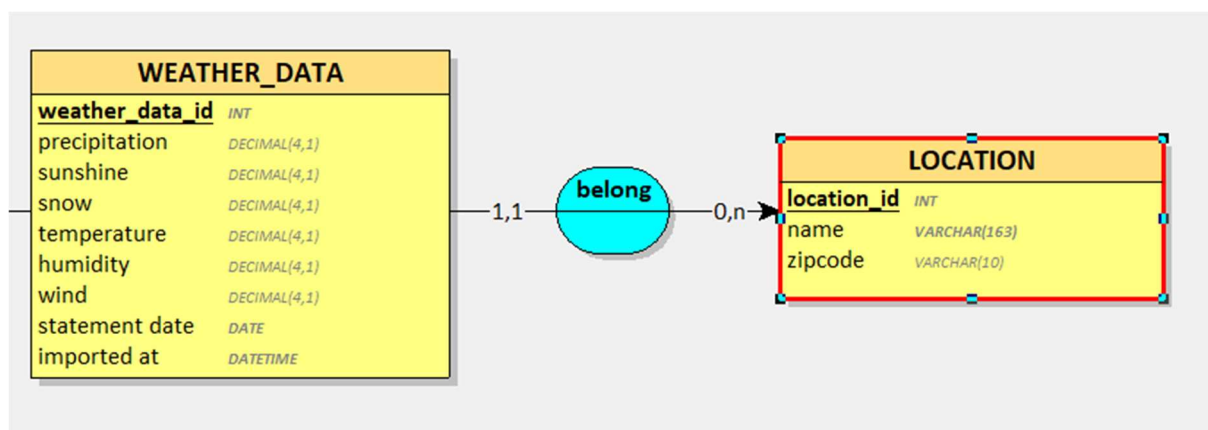


Figure 18 Relation WEATHER_DATA to LOCATION

Cardinalité :

- Un set de données météorologiques est associé à un seul lieu uniquement.
- Un lieu peut avoir 0 à un nombre N de données météorologiques qui lui y est liée.

Relation :

La relation exprime l'appartenance, l'association entre un set de données météorologiques et un lieu.

MLD :

La table t_weather_data prend donc en clé étrangère l'identifiant du lieux, illustré ci-dessous en vert.

t_weather_data	
<u>weather_data_id</u>	INT
precipitation	DECIMAL(4,1)
sunshine	DECIMAL(4,1)
snow	DECIMAL(4,1)
temperature	DECIMAL(4,1)
humidity	DECIMAL(4,1)
wind	DECIMAL(4,1)
statement_date	DATE
imported at	DATETIME
location_fk	INT
user_fk	INT

Figure 19 FK LOCATION t_weather_data

3.3 Conception des tests

Comme présenté en point [1.5.6](#) les tests effectués seront de type fonctionnel et use case. Après chaque implémentation de nouvelle fonctionnalité, elle sera testée directement par la candidate, dans l'environnement de développement.

A la fin du projet, un test général est réalisé sur l'ensemble des fonctionnalités et du fonctionnement du site. Dans la mesure du possible, il faut que ce soit une autre personne qui le fasse afin d'éprouver l'implémentation d'un autre point de vue.

Ci-dessous un tableau de tests qui est repris au point [5.1](#) avec le résultat en pourcentage de réussite.

N°	Nom de la catégorie	Types de tests	Procédure	Résultats attendus
1	Formulaire	Champs obligatoires	Remplir un formulaire en omettant des champs qui sont obligatoires	Le formulaire ne doit pas se soumettre et envoyer un message d'erreur à l'utilisateur
2	Formulaire	Données invalides	Remplir les champs avec des caractères qui sortent des règles Par ex : Mettre des chiffres dans le champ Nom	Le formulaire ne doit pas se soumettre et envoyer un message d'erreur à l'utilisateur
3	Formulaire	Valeurs extrêmes	Remplir les champs avec des caractères spéciaux. Par exemple : ?, !, %, &, ' , ,	Le formulaire ne doit pas se soumettre et envoyer un message d'erreur à l'utilisateur
4	Formulaire	Accents	Remplir les champs avec des accents	Les champs Nom et Prénom sont les seuls à accepter les accents. Les autres champs refusent la soumission du formulaire et renvoie un message d'erreur à l'utilisateur.
5	Formulaire	Espace et apostrophe	Remplir les champs avec un espace et/ou une apostrophe	Les champs Nom et Prénom sont les seuls à accepter les espaces ou une apostrophe. Les autres champs refusent la soumission du formulaire et renvoie un message d'erreur à l'utilisateur.

N°	Nom de la catégorie	Types de tests	Procédure	Résultats attendus
6	Formulaire	Email	Remplir le champ email avec des valeurs erronées	Le champ email précise qu'il doit compter un arobase et un '.' Dans le cas contraire, il refuse la soumission du formulaire et renvoie un message à l'utilisateur.
7	Liens	Validation		L'utilisateur reçoit un email avec un lien de validation en français
8	Liens	Réinitialisation	Demander un lien de réinitialisation du mot de passe	L'utilisateur reçoit un email avec un lien de réinitialisation du mot de passe en français
8	Base de données	Requête fonctionnelle	Vérifier que les soumissions de formulaires inscrivent les données dans la base de données.	En cas d'échec ou de réussite de la requête, l'utilisateur en est informé par un message.
9	Importation	Fichier CSV	Importer un fichier d'un autre type de CSV	Le fichier ne doit pas être accepté par le formulaire d'import et l'utilisateur est informé de l'échec avec un message.
10	Importation et base de données	Import des données	Importer un fichier CSV avec les mauvaises colonnes ou types de données qui ne correspondent pas	L'inscription dans la base de données ne doit pas s'effectuer et l'utilisateur doit être informé que les données ne sont pas conformes.
11	Graphiques	Catégories	Choisir plus d'une catégorie	Les graphiques ne doivent afficher qu'une. L'utilisateur est informé par un message d'erreur.
12	Sécurité	Utilisateur anonyme	L'utilisateur anonyme n'a pas d'accès aux fonctionnalités des autres utilisateurs.	L'utilisateur anonyme n'a pas d'accès aux fonctionnalités des autres utilisateurs.
13	Sécurité	Email non validé	Tenter de se connecter sans avoir validé son email	L'utilisateur ne peut pas se logger sans avoir validé son email et est informé par un message approprié.

N°	Nom de la catégorie	Types de tests	Procédure	Résultats attendus
14	Sécurité	Contributeurs	Les contributeurs ont des accès restreints aux fonctionnalités.	Les contributeurs ont des accès restreints aux fonctionnalités.
15	Sécurité	Administrateur	L'administrateur a une gestion complète du site. Il a un accès aux données des contributeurs mais sans possibilités de les modifier sauf l'inactivité.	L'administrateur a une gestion complète du site. Il a un accès aux données des contributeurs mais sans possibilités de les modifier sauf l'inactivité.
16	Barre de recherche	Termes	Entrer des termes inexistants ou hors sujet. Par exemple : Gratin	L'utilisateur est informé que le terme n'existe pas
17	Barre de recherche	Casse	Entrer des termes existants sous plusieurs formes	La recherche ignore la casse pour une expérience facilitée
18	Barre de recherche	Autocomplétion		La recherche propose une autocomplétion à l'utilisateur
19	Navigation	Dynamique		La navigation est dynamique, les boutons et les liens emmènent l'utilisateur sur la bonne page

Tableau 6 Tableau de tests



3.4 Planification détaillée

La planification détaillée reprend les bases de la planification initiale. Cependant elle est corrigée en fonction des retards pris sur la planification initiale et est conforme à la réalité, ainsi que ce qui est présent sur iceScrum. Par ailleurs le total des heures avec cette planification détaillée correspond à 86,9h.

3.4.1 Sprint 1

Backlog et tâches du Sprint 1 du 13 mai au 22 mai 2024

Date	Feature	Backlog	Tâches	Durée estimée en heures
13 mai 2024	Documentation	RDV Experts - CdP	RDV lancement du TPI avec l'expert 1 : M. Nicolas Borboën	1
			RDV CdP pour le cahier des charges et discussion	1
		Planification initiale	Créer le template de la planification	0.75
			Répartir les features	0.33
			Décompte final	0.75
			Envoi de la planification initiale	0.25
14 mai 2024	Documentation	Planification initiale	Créer toutes les stories et les tâches associées	3
		Rapport Sprint 1 mardi 14	Rédaction du rapport	1
	Analyse	Diagramme de flux	Créer un diagramme de flux	2
15 mai 2024	Analyse	Maquettes	Maquette Contributeur	1.5
16 mai 2024	Analyse	Maquettes	Maquette Accueil	2
			Maquette Admin	1
			Discussion CdP	0.5
		MCD – MLD	Créer le MCD puis le MLD	1.5
		Dictionnaire des données	Créer un dictionnaire des données	2
		RFC et bonnes pratiques	Recherches sur les RFC	0.5
Recherches sur les bonnes pratiques	0.5			
17 mai 2024	Analyse	Stratégies de tests	Spécifier la stratégie de tests	0.33
			Créer le tableau des tests	0.75



	Documentation	SWOT	Effectuer un SWOT	2
		Rapport du sprint 1 vendredi 17 mai	Rédaction du rapport	2
			Envoi du rapport	0.25
		JDTV	Mettre à jour le journal de travail	0.25
		Planification détaillée	Créer le template pour la planification détaillée	0.5
			Mettre à jour la planification détaillée	1.25
21 mai 2024	Implémentations	Implémentations	Installer l'environnement de travail	0.5
	Base de données	Fichiers de migrations	Préparer les fichiers de migrations Laravel	2
			Tests sur les fichiers de migrations dans la base de données	1
		Models	Models : créer et modifier si nécessaire les modèles User, WeatherData et Location	1
	Inscription – Mots de passes – Connexion	Views – Inscription	Formulaire d'inscription	1
	Documentation	Rapport Sprint 1	Rédaction du rapport et envoi avec JDTV	1.25
22 mai 2024	Documentation	Sprint Retrospective	RDV CdP	1
Total des heures estimées du Sprint 1				34.91



3.4.2 Sprint 2

Backlog et tâches associées pour le Sprint 2 du 22 mai au 27 mai 2024.

La date du 22 mai se chevauche sur les Sprints 1 et 2.

Date	Feature	Backlog	Tâches	Durée estimée en heures
22 mai 2024	Inscription – Mots de passes – Connexion	Views – Inscription	Lien de validation	2
			Formulaire de connexion	0.75
		Controllers – Inscription	Controllers	2
			Request	1
23 mai 2024	Documentation	RDV Expert 2	Jeudi 23 mai : Expert 2	1
	Inscription – Mots de passes – Connexion	Gestion des mots de passes	Créer le formulaire de réinitialisation	0.75
			Ajouter le lien de réinitialisation	1.25
		Tests de cas	Test sur l'inscription	0.33
			Test l'envoi d'email	0.33
			Test la réinitialisation du mot de passe	0.33
	Recherches et Graphiques	Page d'accueil	View page d'accueil	1.5
			Barre de navigation	0.75
24 mai 2024	Recherches et Graphiques	Page d'accueil	Barre de recherche	1.5
			Graphique aléatoire	2
			Créer la vue pour la combinaison	1.5
		Graphique	Mettre en place les fonctions qui gèrent l'utilisation des graphiques	2
	Documentation	Rapport Sprint 2 vendredi 24 mai	Rédaction du rapport	0.5
			Envoi du rapport	0.25
27 mai 2024	Documentation	Sprint Retrospective	RDV CdP	1
Total des heures estimée pour le Sprint 2				18.74



3.4.3 Sprint 3

Backlog et tâches associées pour le Sprint 2 du 27 mai au 03 juin 2024.

La date du 27 mai se chevauche sur les Sprints 2 et 3.

Date	Feature	Backlog	Tâches associées	Durée estimée en heures
27 mai 2024	Gestion CRUD – Contributeurs	Views – Contributeurs	Créer la page d'accueil des contributeurs	1
			Créer la view Gérer mon profil	0.5
			Créer la view Gérer mes données	1
		Routes – CRUD	Mettre en place les routes pour le CRUD	0.5
		Controllers – Contributeurs	Implémenter toutes les fonctions du CRUD	2
28 mai 2024	Importation CSV	Importer un fichier CSV	Ajouter à la page d'accueil un formulaire d'import	0.75
			Implémenter les fonctions liées à l'import	1.5
	Gestion CRUD – Contributeurs	Tests de cas d'utilisation – Contributeurs	Gestion du profil personnel	0.5
	RUD – Admin	View – Admin	Modifier la page d'accueil contributeur pour les admins	1
			Créer la view du RUD des contributeurs	1
	Documentation	Route – RUD	Mettre en place les routes pour le RUD	0.5
		Rapport Sprint 2 mardi 28 m	Réaction du rapport, JDTV et envoi	1.25
29 mai 2024	RUD - Admin	Controllers – Admin	Intégrer le lien de réinitialisation du mot de passe dans un email envoyé par l'administrateur	1.5
			Fonctions du RUD	1
30 mai 2024	RUD – Admin	Tests de cas d'utilisation – Admin	Gestion du RUD	0.5
	Tests globaux	Tester de l'application	Reprendre la stratégie de tests et effectuer tous les tests	3
			Corriger les erreurs	1
	Documentation	Guide d'utilisation du site	Créer le guide d'utilisation du site en respectant le CDC	1.5
	Déploiement	Recherche sur le déploiement du site	Effectuer des recherches sur comment déployer le site	2
31 mai 2024	Déploiement	Déployer le site	Tester le déploiement du site	2
	Documentation	Guide de mise en service	Rédiger le guide de mise en service du site en respectant le CDC	1
		Rapport final vendredi 31	Terminer la rédaction du rapport	2



			Envoi	1
	Tests globaux	Tester l'application	Si possible faire tester l'application à un tiers	1
03 juin 2024	Documentation	Sprint Retrospective finale	RDV CdP	1
		Révision du code	Vérification des pages de codes	2
		Rapport final	Relecture du rapport et corrections éventuelles, impression et thermocollage	2
		Journal de travail	Révision du journal du travail	0.75
		Archives des livrables	Préparer les livrables du CDC	0.5
	Rendu	Rendre le projet		0.5
Total des heures estimées pour le Sprint 3				33.25

4 Réalisation

4.1 Dossier de Réalisation

4.1.1 Outils utilisés

- Laravel 10

```
py73spe@INF-A13-M202 MINGW64 /d/TPI/TPI-DataVerse (main)
$ composer create-project laravel/laravel:^10.0 dataverse
```

Figure 20 Installation laravel 10

- TailwindCss²

```
py73spe@INF-A13-M202 MINGW64 /d/TPI/TPI-DataVerse/dataverse (main)
$ npm install -D tailwindcss postcss autoprefixer

added 138 packages, and audited 139 packages in 16s

36 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

py73spe@INF-A13-M202 MINGW64 /d/TPI/TPI-DataVerse/dataverse (main)
$ npx tailwindcss init -p

Created Tailwind CSS config file: tailwind.config.js
Created PostCSS config file: postcss.config.js
```

Figure 21 Installation TailwindCss

- Laravel-Debugbar

```
py73spe@INF-A13-M202 MINGW64 /d/TPI/TPI-DataVerse/dataverse (main)
$ composer require barryvdh/laravel-debugbar --dev
```

Figure 22 Installation DebugBar

- Laravel Charts

```
py73spe@INF-A13-M202 MINGW64 /d/TPI/TPI-DataVerse/dataverse (main)
$ composer require consoleTVs/charts:6.*

py73spe@INF-A13-M202 MINGW64 /d/TPI/TPI-DataVerse/dataverse (main)
$ php artisan vendor:publish --tag=charts_config
```

Figure 23 Installation Laravel Charts

² *TailwindCSS peut être installé de plusieurs manières différentes. Ici, il est important d'avoir NodeJs préalablement installé sur la machine. TailwindCss peut être aussi utilisé de manière plus simple, mais moins complète en intégrant dans les fichiers html des liens de script JavaScript dans la partie « Head ».

4.1.2 Configuration de l'environnement

Pour que l'environnement de Laravel soit adapté au projet, il est nécessaire de configurer le fichier correspondant, le fichier «.env». Ce fichier compte trois modifications principales au départ du projet.

Le nom de l'App est changé pour qu'il corresponde aux besoins. De base, la variable APP_DEBUG est initialisée à true. Il faut tout de même y être attentif car autrement le composant « debug-bar » installé dans ce projet ne fonctionnera pas.

#Changer le nom de l'APP**#S'assurer que le APP_DEBUG est à true pour la debug-bar**

```
APP_NAME=DataVerse
APP_ENV=local
APP_KEY=base64:shX6yZjbZugLRFVXR4C/vwDiWm0qrSY9pxQR6gvl8n0=
APP_DEBUG=true
APP_URL=http://localhost
```

Les variables d'environnement de la base de données doivent être également modifiées en fonction de ce qui est utilisé. Dans le cadre du projet, les connexions à la base de données mysql se font au travers de Docker avec phpMyAdmin.

#Modifier les données de connexion fournie par Docker**#Modifier le nom de la base de données et les identifiants de phpMyAdmin**

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=6033
DB_DATABASE=dataverse
DB_USERNAME=root
DB_PASSWORD=root
```

Enfin la troisième modification principale s'effectue dans la partie environnement qui gère les emails. Dans un cadre de production, il est courant d'avoir recours à une entité externe qui reçoit ces emails étant donné que les utilisateurs ne sont pas réels. Pour ce projet, il s'agit de Mailtrap.io.

#Modifier les informations selon Mailtrap.io**#FROM_ADDRESS : Selon besoin de l'app**

```
MAIL_MAILER=smtp
MAIL_HOST=sandbox.smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=4865ef27d35920
MAIL_PASSWORD=a0313ed5437ea6
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS="no_reply@dataverse.ch"
MAIL_FROM_NAME="${APP_NAME}"
```

4.1.3 Modèle physique de données

Les fichiers de migrations représentent ensemble le modèle physique de données.

Ci-dessous les extraits de code de ces fichiers pour la création des tables.

Laravel implémente de base une colonne de timestamps dans toutes les tables. Elles ne sont pas présente dans le MLD du projet car ce sont des particularités de ce framework.

Table USERS

```
Schema::create('users', function (Blueprint $table) {
    $table->id();
    $table->string('lastname', 80);
    $table->string('firstname', 80);
    $table->string('email', 320)->unique();
    $table->timestamp('email_verified_at')->nullable();
    $table->string('password');
    $table->boolean('is_admin')->default('0');
    $table->boolean('is_activ')->default('1');
    $table->rememberToken();
    $table->timestamps();
});
```

Table WEATHER_DATAS

```
Schema::create('weather_datas', function (Blueprint $table) {
    $table->id();
    $table->decimal('precipitation', total:4, places: 1)->
>nullable();
    $table->decimal('sunshine', total:4, places: 1)->nullable();
    $table->decimal('snow', total:4, places: 1)->nullable();
    $table->decimal('temperature', total:4, places: 1)->
>nullable();
    $table->decimal('humidity', total:4, places: 1)->nullable();
    $table->decimal('wind', total:4, places: 1)->nullable();
    $table->date('statement_date');
    $table->timestamp('imported_at')->useCurrent();
    $table->timestamps();
    $table->foreignIdFor(User::class)->constrained()->
>cascadeOnDelete();
});
```

Table LOCATIONS

```
Schema::create('locations', function (Blueprint $table) {  
    $table->id();  
    $table->string('name', 163)->unique();  
    $table->string('zipcode', 10)->unique()->nullable();  
    $table->timestamps();  
});
```

4.1.4 Arborescence

Comme évoqué plusieurs fois à la lecture de ce rapport, le projet a été construit au travers de Laravel. C'est un framework de php, évolutif et extrêmement populaire. La version 10 est utilisée dans ce travail personnel individuel, la version 11 étant sortie le 12 mars de cette année. La candidate ayant déjà commencé à s'entraîner sur la version 10, il n'a pas été jugé opportun de changer de version en cours d'apprentissage.

Laravel est construit sur un fonctionnement de type MVC, c'est-à-dire Model – View – Controller.

Lors de la création d'un projet Laravel, l'arborescence des fichiers est déjà mise en place et dans la multitude de fichiers, seuls quelques dossiers sont couramment utilisés par le programmeur qui ne se sert du framework que de manière basique. Cette arborescence est explicitée de manière succincte ci-dessous avec quelques exemples illustratifs.

Les branches principales d'un projet sont au nombre de 11. Il y a également plusieurs fichiers indépendants.

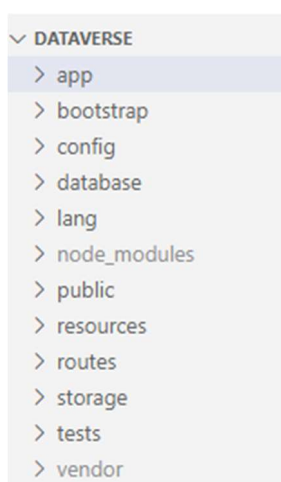


Figure 24 Arborescence principale

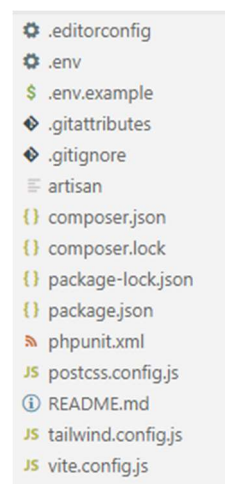


Figure 25 Fichiers principaux

Les dossiers app, database, lang, resources et routes sont les dossiers qui sont majoritairement utilisés dans ce projet.

Dossier app

Il contient plusieurs sous-dossiers, seuls ceux utilisés sont présentés.

- Chart contient les classes pour les graphiques
- Http contient les Controllers et les Request.
- Models contient tous les modèles du projet.

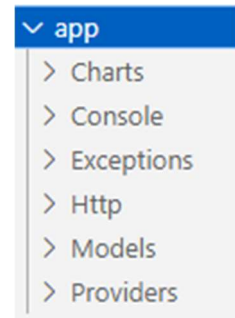


Figure 26 App-Arborescence

Dossier database

Comme son nom l'indique, il contient les sous-dossiers et fichiers en lien avec la base de données.

- Factories contient les classes pour générer des données à remplir dans la base de données
- Migrations contient tous les fichiers de migrations
- Seeders contient les classes qui remplissent la base de données. Ces classes peuvent être alliées avec celles de factories.

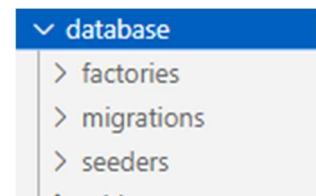


Figure 27 Database - Arborescence

Dossier lang

Le dossier lang n'est pas de base dans un projet Laravel. Il peut être importé par la suite. Dans ce projet, il a été utilisé afin de traduire les messages destinés à l'utilisateur.



Figure 28 Lang - Arborescence

Dossier resources

Le dossier ressources comprend les fichiers en lien avec les Views et l'affichage. Il est nécessaire de souligner que si une feuille de style CSS est utilisée dans le projet, elle n'est pas stockée dans cette partie mais dans la partie « public ».

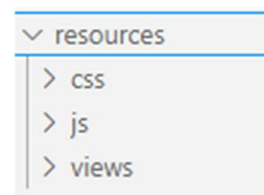


Figure 29 Resource - Arborescence

Le sous-dossiers views englobe toutes les vues et peut être divisé selon les besoins, comme c'est le cas dans ce projet. Les views sont nommées nom.blade.php. Blade est un template inclus dans Laravel qui permet de simplifier le HTML mélangé au PHP.

Exemple de code mixte dans une view

```
@if (session('success') || session('message')) // PHP
<div role="alert"
    class="relative flex w-full px-4 py-4 text-base text-gray-900 rounded-lg
font-regular bg-gray-900/10"> //HTML
    <div class="mr-12">
        <span>{{ session('success') ? session('message') }}</span>
    </div> ...
```

Dossier routes

Une des particularités de Laravel sont les routes. Ce système permet de gérer les demande http entrantes de type GET, POST, PUT ou DELETE. Elles redirigent vers les controllers spécifiques aux demandes. Ces routes sont définies, pour une application web, dans le fichier spécifique web.php. Elles peuvent être nommées, contenir des paramètres, être simple ou complexes.

Exemple route simple

```
/*Route de la page home du site */  
Route::get('/home', [HomeController::class, 'home'])->name('home');
```

Exemple route complexe

```
/*Route qui envoie l'email de réinitialisation COPIE DOC LARAVEL */  
Route::post('/forgot-password', function (Request $request) {  
    $request->validate(['email' => 'required|email']);  
  
    $status = Password::sendResetLink(  
        $request->only('email')  
    );  
  
    return $status === Password::RESET_LINK_SENT  
        ? back()->with(['status' => __($status)])  
        : back()->withErrors(['email' => __($status)]);  
})->middleware('guest')->name('password.email');
```

4.2 Etapes de réalisations

Les différentes étapes de réalisations décrites dans ce chapitre sont dans un ordre chronologique, bien que parfois il a été nécessaire de revenir en arrière, d'apporter des changements et des modifications.

Les sous-chapitres décrivent les points clés des étapes et non pas tout le code, qui lui se trouve en annexe du présent rapport. Les choix et décisions sont expliquées au mieux.

4.2.1 Migrations & Models

La première étape lors de l'implémentation dans le projet a été de préparer les fichiers de migrations pour la base de données ainsi que les modèles des différentes tables.

Les fichiers de migrations sont intégrés dans la base de données par ordre alphabétique, c'est pourquoi il est nécessaire d'être attentif à l'ordre de création des fichiers. Par ailleurs, leur nom va également permettre d'avoir des fonctions préremplies différentes.

Par exemple la commande :

```
php artisan make:migration create_locations_table
```

va permettre d'avoir une migration avec une fonction de création de table inscrite ainsi :

```
Schema::create('locations', function (Blueprint $table) {  
    $table->id();  
    $table->timestamps();  
});
```

A la différence de la commande :

```
php artisan make:migration  
add_foreign_key_location_to_weather_datas_table
```

Celle-là va créer directement un fichier de migration qui permettra d'effectuer des changements sur les tables préexistantes.

```
Schema::table('weather_datas', function (Blueprint $table) {  
    // intégrer les modifications nécessaires  
});
```

Il est possible de créer les clés étrangères et les tables de pivots directement dans les fichiers de migration de type create avec ce type de propriétés.

```
$table->foreignIdFor(User::class)->constrained()->cascadeOnDelete();
```

Cependant, étant donné que Laravel effectue ses migrations par ordre alphabétique, il est parfois impossible de générer une clé étrangère au même moment que la création de la table car les tables reliées devraient être créées en même temps pour que la clé étrangère puisse être produite.

Au niveau des modèles, ils doivent contenir toutes les variables protégées, dites « fillables ». Elles font références aux propriétés des tables.

```
protected $fillable =  
[  
    'precipitation',  
    'sunshine',  
    'snow',  
    'temperature',  
    'humidity',  
    'wind',  
    'statement_date',  
    'imported_at',  
    'user_id',  
    'location_id'  
];
```

Dans le cas, une propriété n'est pas ajoutée dans ces variables, lors d'une interaction avec la base de données par l'application, typiquement comme lors d'un CRUD, cela va générer une erreur.

Les modèles comprennent également toutes les fonctions nécessaires. Dans le cas de clé étrangères ou de tables de pivots, c'est dans cette partie qu'on retrouve les méthodes qui y sont associées.

Exemple de fonction avec clé étrangère

```
/**
 * Fonction protégée correspondant à la liaison entre la table
weather_data et la table user (fk)
 * Permet de retrouver l'utilisateur qui a inscrit les données
météorologiques
 */
protected function users()
{
    return $this->belongsTo(User::class, 'user_id');
}
```

Exemple de fonction standard

```
/**
 * Fonction servant à récupérer le nom et le prénom de l'utilisateur
 *
 */
public function fullName()
{
    return $this->firstname . ' ' . $this->lastname;
}
```

4.2.2 Views Blades, Routes & Controllers

Une fois les migrations fonctionnelles et effectuées, la deuxième étape est de préparer les views principales, les routes initiales et les controllers.

Dans ce projet, une view base.blade.php est générée. Elle comprend les éléments visuels qui se retrouvent sur toutes les pages web et elle contient également les références html essentiels. Toutes les autres vues vont en découler. Grâce aux balises « @include » et « @yield », il est possible d'intégrer des vues à d'autres et d'ainsi avoir des fichiers plus petits et lisibles.

Les routes et les controllers sont intimement liés. Comme démontré dans le chapitre **4.1.4**, les routes reprennent les requêtes http et les envoie aux controllers concernés.

Les controllers comprennent toutes les méthodes nécessaires au fonctionnement du site web. Elles peuvent être très simples, comme de ne retourner qu'une vue pour l'affichage ou alors être complexes.

Exemple méthode moyenne

```
private function getAvailableWeatherDatas($locationId)
{
    //Tableau contenant toutes les données météo
    $weatherdatas = ['precipitation', 'sunshine', 'snow',
'temperature', 'humidity', 'wind' ];

    $availableDatas = [];
    //Recherche dans les données météo les valeurs qui ne sont pas à
null
    //Insértion dans un autre tableau les données qui sont
utilisables.
    foreach ($weatherdatas as $data)
    {
        $count = WeatherData::where('location_id', $locationId)
            ->whereNotNull($data)
            ->count();

        $availableDatas[$data] = $count;
    }
    return $availableDatas;
}
```

Exemple de méthode simple

```
/**Méthode affichant la page du contributeur pour l'administrateur
 * Retourne une vue
 */
public function userSetting($id)
{
    $user = User::find($id);

    return view('admin.settingUser', ['user' => $user]);
}
```

En résumé, lors de la deuxième étape, chaque controller nécessaire a été généré avec des méthodes, même vides, à l'intérieur, pour que les routes fonctionnent et ne génèrent pas d'erreur. Les views associées peuvent également ne pas avoir de contenu, l'idée étant de créer un chemin de navigation primaire.

4.2.3 Request

Après avoir construit les premières routes, views et contrôleurs, ce sont les fichiers de type Request qui ont été produits, dans l'optique de la validation des futurs formulaires.

Les classes Request sont des classes spécifiques à la validation des champs de formulaires. Elles contiennent deux méthodes pré-implémentées par Laravel.

```
/*Détermine si l'utilisateur a le droit d'effectuer cette requête*/
public function authorize(): bool
{
    return true;
}
```

Cette première méthode retourne de base « false », il est primordial de la mettre à « true ». Sans cela, l'utilisateur ne peut pas effectuer la requête du formulaire et est envoyé sur une page d'erreur 403, action non-autorisée.

La deuxième méthode permet d'inscrire toutes les règles de validation des champs de formulaires.

```
/**
 * Méthode de validation pour les requêtes.
 * Contient toutes les règles de validations des champs du formulaire
d'enregistrement
 *
 * @return array<string,
\Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
 */
public function rules(): array
{
    return [
        'lastname' => ['required', 'string', 'regex:/^[a-zA-ZÀ-ÿ\-\']
]+$/' ],
        'firstname' => ['required', 'string', 'regex:/^[a-zA-ZÀ-ÿ\-\']
]+$/' ],
        'email'=> ['required','email', 'regex:/^[-0-9a-zA-Z.+\_]+\@[-0-
9a-zA-Z.+\_]+\.[a-zA-Z]{2,4}$/'],
        'password'=> ['required', 'min:14'],
    ];
}
```

Les éléments dans le tableau sont les règles de validation, ils peuvent être de beaucoup de types différents selon les besoins ; il est aussi possible d'y intégrer les regex pour des champs spécifiques.

4.2.4 Inscriptions – Connexion – Gestion de mot de passes

Dans un troisième temps, toute la partie permettant à un utilisateur anonyme de s'inscrire, puis de se connecter a été mise en place.

Pour répondre aux fonctionnalités techniques exigées pour le projet et représenter au mieux les maquettes, l'utilisateur accède à travers un onglet sur la view home, à un formulaire de connexion. Cette view contient deux liens ; un pour l'inscription et le deuxième pour la réinitialisation du mot de passe.

Les formulaires de connexion et inscription sont gérés par des controllers individuels avec des méthodes propres.

Au niveau de l'inscription plus spécifiquement, après la validation du formulaire par le fichier Request approprié, ce sont des routes qui se chargent d'envoyer un lien de validation par email. Cette fonctionnalité spécifique et son fonctionnement sont disponibles directement dans la documentation de Laravel et ont été repris tels quels dans le projet (<https://laravel.com/docs/10.x/verification#main-content>).

```
/*Route qui envoie l'email de vérification COPIE DOC LARAVEL*/
Route::post('/email/verification-notification', function (Request
$request)
{
    $request->user()->sendEmailVerificationNotification();
    return back()->with('message', 'Verification link sent!');
})->middleware('auth')->name('verification.send');

/* Route qui vérifie si l'email est validé COPIE DOC LARAVEL */
Route::get('/email/verify/{id}/{hash}',
[VerificationController::class, 'verify'])
->middleware('signed')->name('verification.verify');
```

Comme on peut le constater à la lecture de cet exemple, ces routes de gestion d'email de vérification, de validation du lien, etc, sont complexes. Elles contiennent directement des méthodes et des requêtes sans passer par un controller ou un fichier Request.

En suivant donc chaque étape documentée par Laravel, il est possible de mettre en place la validation de l'email sans avoir besoin de créer soi-même les méthodes appropriées.

Concernant la réinitialisation du mot de passe, Laravel propose, tout comme pour le lien de vérification de l'email, une marche à suivre explicite et détaillée dans sa documentation (<https://laravel.com/docs/11.x/passwords#main-content>). Etant donné que la candidate n'avait jamais eu à implémenter cette fonctionnalité auparavant, c'est une reprise de la documentation qui a été mise en place.

Les routes gèrent également toutes les méthodes pour la réinitialisation de mot de passe.

```
/*Route pour la vue de réinitialisation du mot de passe COPIE DOC
LARAVEL*/
Route::get('/forgot-password', function () {
    return view('auth.forgot-password');
})->middleware('guest')->name('password.request');

/*Route qui envoie l'email de réinitialisation COPIE DOC LARAVEL */
Route::post('/forgot-password', function (Request $request) {
    $request->validate(['email' => 'required|email']);
```

```

$status = Password::sendResetLink(
    $request->only('email')
);
return $status === Password::RESET_LINK_SENT
    ? back()->with(['status' => __($status)])
    : back()->withErrors(['email' => __($status)]);
}->middleware('guest')->name('password.email');

/*Route pour le formulaire de reset du mot de passe COPIE DOC LARAVEL */
Route::get('/reset-password/{token}', function (string $token) {
    return view('auth.reset-password', ['token' => $token]);
})->middleware('guest')->name('password.reset');

/*Route qui modifie le mot de passe dans la base de donnée COPIE DOC
LARAVEL */
Route::post('/reset-password', function (Request $request) {
    $request->validate([
        'token' => 'required',
        'email' => 'required|email',
        'password' => 'required|min:14|confirmed',
    ]);
    $status = Password::reset(
        $request->only('email', 'password', 'password_confirmation',
        'token'),
        function (User $user, string $password) {
            $user->forceFill([
                'password' => Hash::make($password)
            ]->setRememberToken(Str::random(60));
            $user->save();

            event(new PasswordReset($user));
        }
    );
    return $status === Password::PASSWORD_RESET
        ? redirect()->route('login')->with('status', __($status))
        : back()->withErrors(['email' => __($status)]);
})->middleware('guest')->name('password.update');

```

L'avantage principal d'utiliser un framework est de pouvoir avoir accès à des fonctionnalités déjà implémentées.

Finalement, seules les views, ci-contre, nécessaires aux différentes routes, ont été créés en suivant strictement les indications de Laravel, notamment pour leurs noms et les dossiers auxquelles elles doivent appartenir.

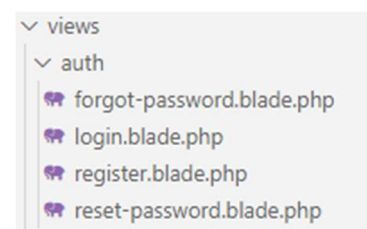


Figure 30 Views - Authentication

4.2.5 Graphique aléatoire de la page d'accueil principale

La page d'accueil principale fait référence à la page des utilisateurs anonymes. La barre de navigation est différente de celle des utilisateurs authentifiés. Elle contient le mot de bienvenue, le graphique aléatoire ainsi que la barre de recherche pour les lieux.

Le graphique aléatoire a pris beaucoup plus de temps que prévu à être implémenté.

En effet, l'idée était de récupérer le mois et l'année en cours, puis d'effectuer un random d'un à dix ans pour trouver l'année correspondante dans la table `weather_datas`. Puis d'effectuer un random sur le type de catégories et d'en créer un graphique.

Cependant Laravel utilise une classe appelée Carbon qui gère toutes les méthodes et propriétés en lien avec les dates. Et ces classes ne permettent pas certaines opérations comme une soustraction, ce qui a généré plusieurs erreurs.

Code générant des erreurs

```
$currentMonth = Carbon::now()->month();
$currentYear = Carbon::now()->year();

//utiliser ->month; & ->year ; permet la soustraction
$randomYear = $currentYear - rand(1,10);
$randomDate = Carbon::create($randomYear, $currentMonth, 1)-
>startOfMonth();
$endOfMonth = $randomDate->endOfMonth();

$weatherData = WeatherData::where('location_id', $location->id)
->whereBetween('statement_date', [$randomDate,
$endOfMonth])
->orderBy('statement_date', 'asc')
->get();
```

En utilisant la propriété `year` et `month` au lieu des méthodes, il était possible d'effectuer la soustraction. Cependant cela générerait une nouvelle erreur en lien directement avec le graphique. Il est généré grâce à l'API Laravel Chart. Cette API simple est basée sur un script JavaScript, et le script prend en charge des tableaux pour générer le graphique et non pas la string obtenue après l'opération de soustraction sur les dates.

Avec toutes ces problèmes pour l'implémentation random de ces dates, il a été décidé de simplifier la méthode au niveau des dates. A la place, cinq dates sont sélectionnées dans un ordre aléatoire, et ordonnées de manière croissante.

```
public function randomWeatherChart($location)
{
    // Vérifier si une localisation a été fournie
    if (!$location) {
        return new NoChartData('Pas de localisation disponible');
    }

    // Déterminer les données météorologiques disponibles pour cette
    localisation et retour si vide
    $availableDatas = $this->getAvailableWeatherDatas($location->id);
    if (empty($availableDatas)) {
        return new NoChartData('Pas de données météo disponible');
    }

    // Sélectionner aléatoirement le type de données météorologiques à afficher
    $randomData = array_rand($availableDatas);

    // Reprendre les données météorologiques pour plusieurs dates aléatoires
    $randomWeatherData = WeatherData::where('location_id', $location->id)
        ->whereNotNull($randomData)
        ->inRandomOrder()
        ->limit(5)
        ->orderBy('statement_date', 'desc')
        ->get();

    // Vérifier si des données météorologiques ont été trouvées
    if ($randomWeatherData->isEmpty()) {
        return new NoChartData('Pas de données météorologiques disponible
pour cette localisation');}

    //Axes X et Y
    $dates = $randomWeatherData->pluck('statement_date')->toArray();
    $weatherDatasDef = $randomWeatherData->pluck($randomData->toArray());

    // Créer le graphique
    $randomChart = new WeatherChart;
    $randomChart->labels($dates);

    $randomChart->dataset(ucfirst($randomData), 'bar', $weatherDatasDef)-
    >backgroundColor('rgb(58, 129, 139)');

    return $randomChart; }
```

Cette méthode a une suite logique simple. Les données sont validées par `getAvailableWeatherDatas()`, méthode privée qui va vérifier que le lieu a des données météo correctes pour un graphique, c'est-à-dire qu'elles ne sont pas vides.

Puis un premier random est effectué sur le type de catégorie de données et le second s'effectuera sur les dates de relevés disponibles pour cette catégorie.

Enfin, comme Laravel Chart demande des tableaux, les variables sont formatées avec `toArray()`. Cela permet d'éviter des erreurs de types lors de la création de graphique.

4.2.6 Barre de recherche

La barre de recherche avait, au départ, été pensée avec autocomplétion. Cependant après plusieurs recherches, il est nécessaire d'utiliser un langage comme JavaScript pour avoir un accès facile, voir possible, à ce genre d'effet. En effet, ce sont des interactions navigateurs-utilisateurs et php ne propose pas ce genre de possibilités.

Bien que le formulaire de recherche soit assez simple à programmer ainsi que la requête dans la base de données, l'implémentation complète a pris le triple du temps prévu.

Etant donné que l'autocomplétion n'était pas possible et pour avoir un aspect similaire de cet effet, il avait été décidé de faire un affichage des lieux, sous forme d'une liste de liens, qui pouvaient ressembler à la recherche effectuée par l'utilisateur. Et c'est cette partie-là qui a fait tripler le temps car l'affichage de la liste ne s'effectuait pas et dans le débogage, elle était vide.

Après plusieurs heures de débogage, relectures de la documentation Laravel, navigations sur les forums, discussions avec l'entourage et « disputes » avec ChatGPT, le code a été redéfini avec une autre approche de la première idée. Il manquait une reprise de la query avec la méthode `get()`.

Le code a été manipulé plus d'une fois afin de toujours améliorer l'expérience utilisateur et la fluidité de la navigation. Ci-dessous est illustré sa version définitive.

```
public function search(Request $request, $view, $randomChartData,
$fromCombi = false) {
    //Récupération de l'ID de l'utilisateur
    //SweatherDatas est utilisé lors du return sur la page home-auth
    if($user = auth()->user()){
        $userId = $user->id;
        $weatherDatas = WeatherData::where('user_id', $userId)
->orderBy('imported_at', 'desc')
->join('locations', 'weather_datas.location_id', '=',
'locations.id')
->limit(5)
->get(['weather_datas.*', 'locations.name']);
    }
```

```

//Reprise de la recherche dans query
$query = $request->input('search');
$location = Location::find($query);

// Recherche de la correspondance exacte
$exactLocation = Location::where('name', $query)->first();

if ($exactLocation) {
    // Rediriger vers la vue combi du lieu trouvé
    return redirect()->route('combi', ['id' => $exactLocation->id]); }
// Si non, continuer avec la recherche partielle
$locations = Location::where('name', 'like', '%' . $query . '%')->get();
// Si la recherche provient de la vue combi, renvoyer à combi avec les résultats partiels
if ($fromCombi) {
    return view('site.combi', [
        'randomChartData' => $randomChartData,
        'search' => $query,
        'locations' => $locations,
        'location' => $locations->first() ?? null,
        'availableYears' => $locations->isNotEmpty()
            ? WeatherData::where('location_id',
$locations->first()->id)
            ->selectRaw('YEAR(statement_date) as
year')
            ->distinct()
            ->pluck('year')
            ->sort()
            ->toArray()
            : [],
        'availableMonths' => $locations->isNotEmpty()
            ? WeatherData::where('location_id',
$locations->first()->id)
            ->selectRaw('MONTH(statement_date) as
month')
            ->distinct()
            ->pluck('month')
            ->sort()
            ->toArray()
            : [],
    ]);
    //Si la recherche provient d'une des vues home
} elseif ($user) {
    return view($view, [
        'randomChartData' => $randomChartData,
        'weatherDatas' => $weatherDatas,
        'location' => $location,

```

```
        'search' => $query,  
        'locations' => $locations,  
    ]);  
}  
else{  
    return view($view, [  
        'randomChartData' => $randomChartData,  
        'location' => $location,  
        'search' => $query,  
        'locations' => $locations,  
    ]);  
}  
}
```

La méthode gère la provenance de la recherche. Comme la barre de recherche est présente sur deux views différentes, il est nécessaire de gérer le résultat différemment en fonction de ce dont la view a besoin pour fonctionner et de si l'utilisateur est authentifié ou non.

4.2.7 Graphiques combinaison

L'utilisateur anonyme et l'utilisateur authentifié contributeur ont la possibilité de créer des combinaisons entre les données et les dates pour générer des graphiques.

Cette étape de réalisation avait déjà été entraînée auparavant et n'aurait pas dû poser autant de problèmes à l'implémentation.

L'utilisateur choisi, à travers un formulaire, quelle catégorie de données météorologiques il veut ainsi que les dates. Au départ, l'intention était de construire une ligne temporelle, avec un curseur. Mais là aussi, comme pour la barre de recherche, il s'agit d'une fonctionnalité de JavaScript. Donc le choix a été fait d'un simple calendrier avec dates de début et date de fin.

Le controller WeatherDataController contient toutes les méthodes pour chaque catégorie de données météorologiques et le controller CombinaisonController les appelle dans sa fonction combinaison. Cependant, pendant un certain temps, une erreur était constamment générée : *Dataset is empty.....* Et en effet, un `dd($combiChart)` démontrait que les labels et le dataset nécessaires étaient des tableaux vides.

Après plusieurs temps de débogage pour comprendre et une relecture ligne par ligne du code, il s'est trouvé que les values de l'input du formulaire et les catégories du switch n'avaient pas les mêmes orthographes, notamment pour « precipitation ».

Après ce réglage, cela a permis de débloquent toutes les problématiques liées à la combinaison de graphiques.

Enfin en dernière étape, une réflexion a été menée autour des dates à choisir par l'utilisateur. Il n'est pas censé savoir quelles sont les mois ou années disponibles dans la table de la base de données. De plus, avec le calendrier, il avait la possibilité d'aller

au-delà de la date actuelle. Cette réflexion a également permis de modifier la méthode du randomWeatherChart qui, au tout départ de la phase de réalisation, n'affichait qu'une seule date.

Afin d'améliorer l'expérience utilisateur et de faciliter une gestion d'erreurs possibles, le choix des dates a été totalement modifié. L'utilisateur peut choisir dans des listes déroulantes les années et les mois disponibles. Cette sélection est reprise, puis traitée dans la méthode combinaison.

La fonction combinaisonChart est très longue à la lecture car elle récupère et traite beaucoup d'éléments à la fois. Pour des questions de structurations et de lisibilité, elle a été divisée en trois parties dont deux sont des méthodes privées. La première est pour la création de graphiques selon la catégorie choisie et la deuxième est pour le return qui comprend énormément de données.

```
public function combinaisonChart($id, Request $request)
{
    $location = Location::findOrFail($id);
// Récupérer les données météorologiques pour ce lieu et cette plage de
dates
    $beginYear = $request->input('begin_year');
    $beginMonth = $request->input('begin_month');
    $endYear = $request->input('end_year');
    $endMonth = $request->input('end_month');
    $category = $request->input('category');
//Validation des catégories pour des questions d'être certain que les
données fournies correspondent à ce qui est attendu
    $validCategories = ['precipitation', 'sunshine', 'snow', 'wind',
'temperature', 'humidity'];
    if (!in_array($category, $validCategories)) {
        $noChartData = new NoChartData('Catégorie invalide');
        return $this->returnCombiView($noChartData, $location);
    }
//Formatage et calcul des dates
    $beginDate = "$beginYear-$beginMonth-01";
    $endDate = date("Y-m-t", strtotime("$endYear-$endMonth-01"));
//Récupération des valeurs choisies par l'utilisateur pour son graphique
    $weatherData = WeatherData::where('location_id', $id)
```

```
->whereBetween('statement_date', [$beginDate, $endDate])
->orderBy('statement_date', 'asc')
->get();

if ($weatherData->isEmpty()) {
    $noChartData = new NoChartData('Pas de données météorologiques
disponible pour cette période');
    return $this->returnCombiView($noChartData, $location);
}

$combiChart = $this->createCombiChart($category, $weatherData);
return $this->returnCombiView($combiChart, $location, $beginYear,
$beginMonth, $endYear, $endMonth, $category); }
```

En substance, la méthode combinaisonChart reprend les informations du formulaire rempli par l'utilisateur, les traite et effectue une recherche dans la base de données.

Ci-dessous est illustrée la fonction privée createCombiChart. Elle reprend les données de la catégorie et des dates et génère un graphique selon la catégorie sélectionnée à travers un switch.

```
private function createCombiChart($category, $weatherData)
{
    switch ($category) {
        case 'precipitation':
            return app(WeatherChartController::class)-
>precipChart($weatherData);
        case 'sunshine':
            return app(WeatherChartController::class)-
>sunshineChart($weatherData);
        case 'snow':
            return app(WeatherChartController::class)-
>snowChart($weatherData);
        case 'wind':
            return app(WeatherChartController::class)-
>windChart($weatherData);
        case 'temperature':
            return app(WeatherChartController::class)-
>tempChart($weatherData);
        case 'humidity':
            return app(WeatherChartController::class)-
>humiChart($weatherData);
        default:
            return new NoChartData('Catégorie invalide');
    }
}
```

4.2.8 Importation de fichier CSV

L'importation de fichier CSV peut s'effectuer si l'utilisateur est authentifié et elle est intégrée à une view spécifique.

L'utilisateur est informé des conditions et de la mise en forme que doit avoir le fichier s'il veut que l'importation soit fonctionnelle. Dans le cas où cela ne respecterait pas les conditions préalables, il est informé par des messages d'erreurs appropriés.

Etant donné que le formulaire permet également, en plus de l'import, l'ajout d'un nouveau lieu si l'utilisateur ne trouve pas ce qu'il veut dans la liste, la méthode gère deux éléments différents.

De plus, une partie de gestion des erreurs supplémentaires qui pourraient être dans le fichier CSV a été ajoutée par après.

```
public function process(ImportRequest $request)
{
1ère partie pour les lieux
// Récupérer les données du formulaire
$newLocationName = $request->input('newLocationName');
$newLocationZip = $request->input('newLocationZip');
$selectedLocationId = $request->input('locationImport');

// Vérifier si aucun lieu n'est sélectionné ou ajouté
if (!$selectedLocationId && (!$newLocationName || !$newLocationZip))
{
    return redirect()->back()->withErrors('Merci de choisir un lieu
dans la liste ou d\'en ajouter un.');
```

```

// Si les champs pour le nouveau lieu sont remplis, créer un nouveau
lieu dans la db
if ($newLocationName && $newLocationZip) {
    $location = Location::create([
        'name' => $newLocationName,
        'zipcode' => $newLocationZip ]);

    // Récupérer l'ID du nouveau lieu créé
    $selectedLocationId = $location->id;
}
// Récupérer id de l'utilisateur
$user = auth()->user();
$userId = $user->id;

2ème partie pour le fichier
$file = $request->file('csv_file');
```

```

// Validation que le fichier a été téléchargé et est un fichier CSV
if ($file && $file->getClientOriginalExtension() == 'csv') {
```



```
// Lecture du contenu du fichier CSV
$contentts = file_get_contents($file->getRealPath());

// Traitement des données CSV et insertion dans la base de données
$rows = explode("\n", $contentts);

// Ignorer la première ligne (en-têtes de colonnes)
$header = array_shift($rows);
$expectedHeaders = ['precipitation', 'sunshine', 'snow',
'temperature', 'humidity', 'wind', 'statement_date'];
$headerColumns = str_getcsv($header);

// Valider les en-têtes
if ($headerColumns !== $expectedHeaders) {
return redirect()->back()->withErrors('Les en-têtes du fichier CSV ne
sont pas corrects.');
```

```
// Ignorer les lignes vides
$nonEmptyRows = array_filter($rows, 'strlen');
if (!empty($nonEmptyRows)) {
    foreach ($nonEmptyRows as $row) {
        $data = str_getcsv($row);

// Validation du nombre de colonnes
        if (count($data) !== 7) {
            return redirect()->back()->withErrors('Le fichier CSV doit contenir
exactement 7 colonnes.');
```

```
// Validation des types de données
        if (!is_numeric(trim($data[0])) || !is_numeric(trim($data[1])) ||
!is_numeric(trim($data[2])) || !is_numeric(trim($data[3])) ||
!is_numeric(trim($data[4])) || !is_numeric(trim($data[5]))
||!strtotime(trim($data[6])))
        {
            return redirect()->back()->withErrors('Les données du fichier CSV ne
sont pas du bon type. Veuillez vérifier les colonnes.');
```

```
// Insérer directement dans la base de données
WeatherData::create(['precipitation' => trim($data[0]),
                    'sunshine' => trim($data[1]),
                    'snow' => trim($data[2]),
                    'temperature' => trim($data[3]),
                    'humidity' => trim($data[4]),
                    'wind' => trim($data[5]),
                    'statement_date' => trim($data[6]),
                    'user_id' => $userId,
                    'location_id' => $selectedLocationId]);
return redirect()->route('import.showForm')->with('success', 'Importation
réussie.');
```

```
        } else {  
            return redirect()->back()->withErrors('Le fichier CSV est vide.');
```

```
        }  
        return redirect()->back()->withErrors('Veuillez sélectionner un fichier  
de format CSV.');
```

En utilisant la méthode `WeatherData::create()`, une modification de la table `weather_datas` a dû être effectuée dans le fichier de migration du projet.

En effet Laravel utilise des timestamps `created_at` et `updated_at` dans ses méthodes en lien avec des requêtes, et ces deux colonnes n'étaient pas intégrées à la table avant. Cependant le MLD construit à travers Looping pour la partie analytique du projet n'a, lui, pas été modifié car cela n'a pas d'incidence sur la logique de la base de données.

4.2.9 CRUD & RUD

Les contributeurs et les administrateurs ont des fonctionnalités supplémentaires comparés à l'utilisateur anonyme. Ils peuvent importer des données, gérer les données qu'ils ont importées et ont une gestion personnelle de leur compte. L'administrateur peut également gérer les contributeurs, en désactivant ou activant leur compte par exemple.

Dans la gestion des données liées à un utilisateur, il a la possibilité de réimporter un fichier pour remplacer celles existantes. La méthode diffère peu de celle décrite dans le point **4.2.8** à la différence qu'au lieu de faire une création, elle gère une update.

```
...  
// Mettre à jour l'enregistrement existant  
    $weatherData->update([  
        'precipitation' => trim($data[0]),  
        'sunshine' => trim($data[1]),  
        'snow' => trim($data[2]),  
        'temperature' => trim($data[3]),  
        'humidity' => trim($data[4]),  
        'wind' => trim($data[5]),  
        'statement_date' => trim($data[6]),  
    ]);  
...
```

L'administrateur peut renvoyer un lien de réinitialisation du mot de passes à un contributeur qui en aurait fait la demande. La méthode qui gère cette fonctionnalité est 100% inspirée de la route utilisée pour l'oubli du mot de passe disponible à la connexion.

Elle contient beaucoup de fonctions de Laravel et a été implémentée avec l'aide de la documentation.

```
public function sendResetLink($id)  
{  
    $user = User::find($id);  
    $status = Password::sendResetLink(['email' => $user->email]);  
  
    return $status === Password::RESET_LINK_SENT  
        ? redirect()->route('user.setting', ['id' => $user->id])-  
        >with('success', 'Lien envoyé à l\'utilisateur')  
        : redirect()->route('user.setting', ['id' => $user->id])-  
        >withErrors(['email' => __($status)]);  
}
```

4.3 Modifications

Des modifications sont apportées tout au long du projet et de son implémentation.

Le tableau ci-dessous reprend les modifications majeures dans le projet que cela soit au niveau de méthodes, de logique de code, de controllers, etc.

Les petites modifications sont toutes disponibles sur le repository gitHub avec les commits exécutés au fur et à mesure du projet.

Date(s)	Fonctionnalités	Raison	Description
23.05.2024	Graphique aléatoire	Affichage non adéquat	L'affichage de base du graphique aléatoire n'étant pas celui escompté, modification de l'idée originelle, discuté avec le chef de projet lors du 2 ^e Sprint Review.
24.05.2024	Barre de recherche et autocomplétion	Autocomplétion impossible sans JavaScript	Modification du fonctionnement de la barre de recherche sans autocomplétion, affichage d'une liste de lieux existants à la place.
27.05.2024	Ligne temporelle	Ligne temporelle qui bouge impossible sans JavaScript	A la place, mise en place d'un petit formulaire qui demande les dates de début et de fin pour les graphiques.
28.05.2024	Importation CSV	View individuelle	Dans les maquettes, l'importation des fichiers devait pouvoir s'effectuer dans la page d'accueil. Cependant cela créé des conflits entre les variables pour les lieux. Choix de mettre une vue individuelle pour l'import.
30.05.2024	Importation CSV	Gestion des erreurs	Etant donné que les erreurs sur le fichier CSV n'étaient pas toutes gérées et/ou qu'elles renvoyaient une erreur non compréhensible pour l'utilisateur, modification de la fonction process en conséquence.
31.05.2024	CRUD données météo	Modifications views	Pour avoir une view combinée dans l'affichage et la modification des données météo, modification du SettDataController et des views concernées.

Tableau 7 Tableau des modifications

5 Tests

Pour effectuer les tests dans un cadre extérieur à l'environnement de développement, un fichier de script SQL est disponible dans le repository de GitHub. Un utilisateur de type « contributeur » ainsi qu'un administrateur sont disponibles. Leurs identifiants sont :

- john@doe.fr – testContributeur14
- admin@admin.ch – testAdministrateur14

Il y a également plusieurs fichiers CSV qui contiennent des erreurs pour faire les différents tests sur les imports.

5.1 Dossier des tests

Le tableau de tests du point 3.3 sont repris dans le tableau ci-dessous avec la(les) dates de tests et le résultat en pourcentage de réussite. Les tests qui n'ont pas eu le résultat escompté ou qui n'ont pas pu être effectués sont relevés dans le sous-chapitre suivant.

L'ensemble de tests et de l'application ont été éprouvés par la candidate et par un tiers externe qui n'a pas de profession dans l'informatique.

N°	Nom de la catégorie	Types de tests	Procédure	Date du test	Résultats en %
1	Formulaire	Champs obligatoires	Remplir un formulaire en omettant des champs qui sont obligatoires	22.05.2024	100
2	Formulaire	Données invalides	Remplir les champs avec des caractères qui sortent des règles Par ex : Mettre des chiffres dans le champ Nom	22.05.2024	100
3	Formulaire	Valeurs extrêmes	Remplir les champs avec des caractères spéciaux. Par exemple : ?, !, %, &, ' , '	22.05.2024	100
4	Formulaire	Accents	Remplir les champs avec des accents	22.05.2024	100
5	Formulaire	Espace et apostrophe	Remplir les champs avec un espace et/ou une apostrophe	22.05.2024	100
6	Formulaire	Email	Remplir le champ email avec des valeurs erronées	22.05.2024	100
7	Liens	Validation		21.05.24	100
8	Liens	Réinitialisation	Demander un lien de réinitialisation du mot de passe	21.05.24	100
8	Base de données	Requête fonctionnelle	Vérifier que les soumissions de formulaires inscrivent les données dans la base de données.	21.05.24	100
9	Importation	Fichier CSV	Importer un fichier d'un autre type de CSV	28.05.2024	100
10	Importation et base de données	Import des données	Importer un fichier CSV avec les mauvaises colonnes ou types de	28.05.2024 30.05.2024	0 100

N°	Nom de la catégorie	Types de tests	Procédure	Date du test	Résultats en %
			données qui ne correspondent pas		
11	Graphiques	Catégories	Choisir plus de deux catégories	27.05.24	100
12	Sécurité	Utilisateur anonyme	L'utilisateur anonyme n'a pas accès aux fonctionnalités des autres.	31.05.2024	100
13	Sécurité	Email non validé	Tenter de se connecter sans avoir validé son email	21.05.24	100
14	Sécurité	Contributeurs	L'utilisateur non-admin n'a pas accès aux fonctionnalités des admins.	31.05.2024	100
15	Sécurité	Administrateur	L'administrateur a un accès complet	31.05.2024	100
16	Barre de recherche	Termes	Entrer des termes inexistants ou hors sujet. Par exemple : Gratin	28.05.2024	100
17	Barre de recherche	Casse	Entrer des termes existants sous plusieurs formes	28.05.2024	100
18	Barre de recherche	Autocomplétion	-	23.05.2024	0
19	Navigation	Dynamique		31.05.2024	100

Tableau 8 Tableau de résultat des tests

5.2 Tests échoués, non réalisés & bugs

Test n°10 : 28.05.2024 → Si le fichier CSV comprend des valeurs supplémentaires ou des colonnes avec de mauvais noms, l'enregistrement s'effectuent quand même dans la base de données. Les données qui ne correspondent pas au type de la valeur, par exemple des lettres ou des caractères spéciaux génèrent une erreur `SQLSTATE[HY000]: General error: 1366 Incorrect decimal value:`

29.05.2024 **EN COURS** : Il faut modifier la fonction process de l'ImportController pour vérifier les valeurs et générer une erreur compréhensible pour l'utilisateur.

30.05.2024 **RESOLU** : Modification pour la fonction process de l'ImportController afin de mieux gérer les différentes erreurs dans le fichier CSV avec des erreurs appropriées pour l'utilisateur.

Test n° 18 : 23.05.2024 → L'autocomplétion n'est pas implémentée car elle demande une structure avec JavaScript. En l'occurrence ce projet ne prend pas en charge l'implémentation de ce langage.

BUG : 02.06.2024 → Lors du test par un tiers externe, l'unicité du mail a été provoqué une erreur illisible pour l'utilisateur.

03.06.2024 **RESOLU** : Modification dans les règles de validation de fichier RegisterRequest, en ajoutant la règle « unique ».

6 Conclusion

6.1 Bilan des fonctionnalités demandées

Le tableau ci-dessous reprend les fonctionnalités décrites dans le point **1.5.3** et définit si elles sont atteintes. Dans le cas où ce n'est pas le cas, la dernière colonne indique une estimation pour la terminer.

UA = utilisateur anonyme – C = contributeur/inscrit – A = Administrateur – T = tous

Fonctionnalité	Qui	Réussi	Echoué	Estimation
Informer du but du site	UA	<input checked="" type="checkbox"/>		
Visualiser les catégories de données	UA-C	<input checked="" type="checkbox"/>		
Visualiser les catégories géographiques	UA-C	<input checked="" type="checkbox"/>		
Rechercher des lieux	UA-C	<input checked="" type="checkbox"/>		
Combiner des graphiques avec des dates	UA-C	<input checked="" type="checkbox"/>		
Inscription sur le site	T	<input checked="" type="checkbox"/>		
Email de validation valable 24h	T	<input checked="" type="checkbox"/>		
Login sur le site	C-A	<input checked="" type="checkbox"/>		
Mot de passe chiffré	C-A	<input checked="" type="checkbox"/>		
Lien de réinitialisation	C-A	<input checked="" type="checkbox"/>		
Gestion du profil personnel	C-A	<input checked="" type="checkbox"/>		
Gestion du mot de passe	C-A	<input checked="" type="checkbox"/>		
Désactivation du compte	C-A	<input checked="" type="checkbox"/>		
Gestion des données météorologiques	C-A	<input checked="" type="checkbox"/>		
Importer des fichiers CSV	C-A	<input checked="" type="checkbox"/>		98%
Désactiver des données météorologiques erronées	A	<input checked="" type="checkbox"/>		
Désactiver les contributeurs	A	<input checked="" type="checkbox"/>		
Maintenir les données météorologiques de tous	A		<input checked="" type="checkbox"/>	95%

L'import de fichiers CSV fonctionne très bien et les erreurs dans le fichier sont gérées. Cependant on peut constater que si l'utilisateur ajoute un nouveau lieu avec le formulaire, il peut en ajouter des déjà existants. Cela crée donc des doublons dans la base de données.

Le maintien des données météorologiques de tous n'a pas été vraiment implémenté, car en soi comme l'administrateur a la possibilité d'accéder à la base de données avec un lien phpMyAdmin, il peut le faire depuis là si besoin. Cela étant, cela ne demanderait pas beaucoup de temps de mettre cette fonctionnalité en place.

6.1.1 Suites à donner

Dans les suites à donner, il serait possible d'ajouter une gestion des données par l'administrateur directement sur l'application web. Cela pourrait être une fonctionnalité plus sécuritaire et qui n'obligerait pas l'administrateur à avoir des connaissances en langage SQL et des compétences de gestion de base de données. Cela permettrait à tout un chacun d'être administrateur du site. Et par là-même, il faudrait ajouter une possibilité de promouvoir un contributeur en administrateur.

Lorsque l'utilisateur fait appel au lien de réinitialisation pour changer son mot de passe, il ne reçoit aucun message de succès ou d'échec. Il faudrait l'implémenter par la suite. Cela n'a pas été mis en place dans ce projet car comme c'est tout gérer via des routes, la candidate ne savait pas exactement comme le faire.

Le fonctionnement et les possibilités de graphiques devraient avoir un encart explicatif. De plus, la gestion des noms identiques et multiples doit être gérées. Des règles de validations ont été ajoutées pour amener une unicité pour un lieu et pour un code postal. Cela règle une partie de la problématique car on ne peut plus ajouter un lieu ou un code postal déjà existant mais ils ne sont pas liés entre eux.

Enfin, si l'utilisateur effectue un changement d'email et/ou de mot de passes avec le formulaire de gestion de données personnelles, il a un message de succès ou des messages d'erreurs. Cependant, il pourrait être plus sécuritaire si ces modifications s'effectuaient avec des emails de validation en sus.

Et pour une amélioration globale du site, il serait très intéressant d'y intégrer du JavaScript pour l'expérience utilisateur-navigateur et pour que le site soit plus attractif et attrayant.

6.2 Bilan de la planification

La planification initiale et la planification finale ont pas mal de différences, pas tant en termes de durée mais surtout en termes de répartitions des tâches. En comptant les trois heures effectuées en sus des 88h, le dépassement n'est pas énorme.

En effet, le rythme de travail prévu a été beaucoup chamboulé par trois fonctionnalités en particulier.

La première est l'implémentation des graphiques. La tâche avait été estimée à environ 5h. Cependant elle a pris environ 2h de plus. Le graphique aléatoire a été plus complexe que prévu au vu des problèmes soulevés dans le point **4.2.5**, tout comme la combinaison des graphiques (point **4.2.7**).

La barre de recherche a également pris beaucoup de retard par rapport à ce qui était prévu. Estimée à 1h30, elle en a finalement pris 4h de plus avec une partie du travail exécutée depuis la maison afin de ne pas perdre trop de temps sur l'ensemble.

Enfin, la troisième tâche qui pris plus de temps a été l'importation des fichiers CSV car la gestion des erreurs n'étaient pas complètes sur la première version.

Le jeudi 30 mai 2024, lors de la mise en route de l'environnement, il s'est révélé que les fichiers du projet Laravel pour ce TPI étaient corrompus et le SSD a décidé de ne plus être accessible sauf avec des identifiants administrateurs de la DGEP. Dès lors, il y a eu

une perte de temps d'1h15 sur la journée. La candidate a décidé de prendre ce temps le soir pour finaliser quelques méthodes qui avaient encore des erreurs.

Dans le cahier des charges et les planifications, il était prévu de déployer le site en ligne. Cependant au vu du retard dans les implémentations et la rédaction des documents, il a été discuté lors du Sprint Review du 22 mai 2024 avec le chef de projet de l'abandon de cette tâche. Par conséquent, le guide de mise en service est lui aussi abandonné.

6.3 Bilan personnel

L'ensemble du projet a été très satisfaisant pour moi. Bien que le sujet du site ne me soit absolument pas familier, pouvoir y effectuer des recherches était instructif et enrichissant. Avoir la possibilité de créer un projet avec Laravel m'a permis d'approfondir mes connaissances sur ce framework, technique qui me tenait à cœur ayant des projets personnels qui me demanderont de l'utiliser.

Au niveau de la planification, je me suis rendu compte qu'il était difficile pour moi d'estimer le temps que je peux passer sur une tâche ou sur une autre. N'ayant pas la même notion du temps que d'autres personnes, il est nécessaire que je porte une attention particulière à ce niveau. L'utilisation d'une application comme IceScrum était très utile car elle m'a permis de découper le projet en micro-tâches, mais cela n'a pas été aidant pour la répartition des heures.

Finalement, n'étant plus dans un contexte de « travail d'entreprise » depuis bientôt trois ans, cela m'a permis de m'auto-évaluer sur mes limites et mes capacités personnelles dans la gestion du stress, des compétences acquises durant la formation et d'appuyer mon choix de changement professionnel.

6.4 Remerciements

Je tiens à remercier toutes les personnes qui ont été présentes durant ce projet, en particulier mon chef de projet, Bertrand Sahli pour sa bienveillance, sa critique constructive et ses encouragements dans mes moments de stress.

Je remercie mes experts pour leur disponibilité, leurs remarques pertinentes et leur compréhension.

Je remercie également ma famille qui a été d'un soutien sans faille durant toute cette formation et pendant ce TPI.

7 Divers

7.1 Journal de travail

En annexe

7.2 Webographie

IMAGE DE PAGE DE GARDE :

<https://www.gettyimages.ch/detail/foto/storm-noa-2023-cloud-map-atlantic-ocean-europe-3d-lizenzfreies-bild/1487001908?et=ciVkdCSoNMHSIWgloJ0Q&referrer=https%3A%2F%2Fwww.gettyimages.fr%2F>

ANALYSE :

https://datavizcatalogue.com/FR/methodes/diagramme_de_flux.html

<https://app.diagrams.net/#Wb!kDOWyRMv0UiFBDRd7oQnTllg5gLTHplEs8XWMCDeITbn2Y8w6Au6RIHzcjYoy5XV%2F01FCA2JNXIYFYAVVTUCBB245V27VVSgKGQ#%7B%22pageId%22%3A%22C5RBs43oDq-KdzZeNtuy%22%7D>

https://fr.wikipedia.org/wiki/Rhoshandiatelly-neshiaunneveshenk_Koyaanfquatsiuty

https://fr.wikipedia.org/wiki/Liste_des_toponymes_les_plus_long#:~:text=Le%20nom%20de%20pays%20e,Nord%20qui%20fait%2042%20lettres.

<https://www.lapresse.ca/actualites/insolite/201309/13/01-4689306-un-nom-de-famille-trop-long-pour-son-permis-de-conduire.php#:~:text=Janice%20%C2%ABLokelani%20Keihanaikukauakahihuliheekahaunaale%20se%20bat%20pour%20faire%20changer%20cela.&text=Les%20documents%20ne%20pouvant%20contenir,et%20sa%20carte%20d'identit%C3%A9.>

https://fr.wikipedia.org/wiki/Code_postal

https://en.wikipedia.org/wiki/List_of_postal_codes

<https://igm.univ-mlv.fr/~dr/XPOSE2011/BDD/fn.html>

<https://stph.scenari-community.org/bdd/0/co/norUC025.html>

<https://learn.microsoft.com/fr-fr/sql/relational-databases/import-export/specify-field-length-by-using-bcp-sql-server?view=sql-server-ver16>

<https://datatracker.ietf.org/doc/html/rfc5321#section-4.5.3>

<https://www.dynamic-mess.com/sql/base-de-donnees-differents-types-de-champ/>

<https://dev.mysql.com/doc/refman/8.0/en/fixed-point-types.html>

https://fr.wikipedia.org/wiki/ISO_8601

<https://www.rfc-editor.org/rfc/rfc3339#section-5>

<https://educrak.com/chapitre-content/64b5d49415301-les-routes-controleurs-et-vues>

RÉALISATION :

<https://laravel.com/docs/10.x>

<https://charts.erik.cat/>

<https://tailwindcss.com/resources>

<https://regex101.com/>

<https://laravel.com/docs/11.x/passwords#main-content>

<https://htmlcolorcodes.com/fr/selecteur-de-couleur/>

<https://www.apprendre-laravel.fr/laraguide/2017-11-26-mettre-a-jour-nos-utilisateurs-avec-eloquent>

<https://laravel.com/docs/11.x/helpers#method-array-random>

<https://carbon.nesbot.com/docs/#api-addsub>

<https://www.youtube.com/watch?v=Y46f5LTAO0Y> / Carbon Et Laravel

Barre de recherche

<https://www.youtube.com/watch?v=R58XZ8pAXoE>

<https://www.youtube.com/watch?app=desktop&v=gwGBo7DDxhA>

<https://laravel.com/docs/10.x/queries#where-clauses>

<https://laravel.com/docs/10.x/collections#method-collect>

<https://laravel.com/docs/10.x/requests#input>

GLOSSAIRE :

<https://www.redhat.com/fr/topics/api/what-are-application-programming-interfaces>

<https://fr.wikipedia.org/wiki/Framework>

<https://www.usabilis.com/ui-design/>

<https://www.guru99.com/fr/use-case-testing.html>

<https://latavernedutesteur.fr/2019/01/10/techniques-basees-sur-les-specifications-7-7-tests-de-cas-dutilisation/#:~:text=Le%20test%20de%20cas%20d,utilisateurs%20ou%20des%20syst%C3%A8mes%20ext%C3%A9rieurs.>

<https://www.free-work.com/fr/tech-it/blog/actualites-informatiques/tests-unitaires-vs-tests-fonctionnels>

<https://www.loadview-testing.com/fr/test-de-charge/#:~:text=Le%20test%20de%20charge%20est,la%20d%C3%A9gradation%20et%20l%C3%A9volutivité%C3%A9.>

<https://www.frenchweb.fr/quest-ce-quune-user-story-en-methode-agile/441373>

<https://dictionnaire.lerobert.com/definition/autocompletion>

<https://www.wildcodeschool.com/fr-fr/blog/differences-backend-frontend-developpement-web#:~:text=Le%20backend%20est%20responsable%20du,les%20besoins%20de%20l'application.>

https://fr.wikipedia.org/wiki/Jeton_d%27acc%C3%A8s

<https://miro.com/fr/diagramme-de-flux/qu-est-ce-qu-un-diagramme-de-flux/>

https://fr.wikipedia.org/wiki/Test_d%27int%C3%A9gration

<https://developer.mozilla.org/fr/docs/Glossary/MVC>

8 Annexes

8.1 Glossaire

API : Application Programming Interface (en) / Interface de programmation d'applications (fr). Outil permettant la transmissions et l'accès des données de façon simplifiée et standardisée.

Autocomplétion : Fonctionnalité qui propose des mots à l'utilisateur à partir des premiers caractères saisis.

Backend : Partie de la programmation de l'application invisible pour l'utilisateur, composants internes d'une application ou d'un logiciel. *En opposition au frontend*

CRUD : Acronyme anglais de Create, Read, Update et Delete. Cela correspond aux différentes possibilités d'opérations avec la base de données pour l'utilisateur.

Diagramme de flux : Représentation graphique qui décrit un processus, un système ou un algorithme informatique dans un ordre séquentiel.

Features : Dans le cadre de la méthodologie de projet AGILE et SCRUM, une feature (fonctionnalité en français) représente une thématique rassemblant plusieurs user stories qui portent sur le même sujet.

Framework : Infrastructure logicielle, ensemble cohérent de composants logiciels structurels servant à créer les fondations d'une architecture. N'est pas équivalent à une bibliothèque logicielle (library).

Frontend : Partie de la programmation de l'application visible et interactive pour l'utilisateur. *En opposition au backend.*

Jeton (de session) : Contient des informations d'identification de sécurité pour une connexion et identifie l'utilisateur.

MVC : Modèle de conception logiciels, séparant la logique métier et l'affichage du logiciel. Le M est pour Model, qui gère la partie de base de données, V pour View qui gère l'affichage et le C de Controller qui achemine les commandes entre les parties M et V.

RUD : Acronyme anglais de Read, Update, Delete. Ce sont les mêmes opérations que le CRUD au sein de la base de données par un utilisateur mais sans la création.

Test de cas d'utilisation : Ce type de test fournit des tests basés sur des scénarios qui simulent l'usage du site web entre l'utilisateur et le système.

Test de charge : Simulation de l'utilisation du site web dans le monde « réel » pour identifier des facteurs de réactivité, de dégradation et d'évolutivité.

Test d'intégration : Interviennent après les tests unitaires, ils vérifient que toutes les unités d'une application fonctionnent ensemble dans n'importe quel environnement.

Test fonctionnel : Test défini par l'ISO-25010 couvrant les tests d'exactitude, de complétude et d'aptitude à l'usage. C'est-à-dire, qu'il a pour but d'assurer que l'application répond aux attentes. Il fait souvent partie des tests d'acceptation.

Test unitaire : Test est écrit dans le but de tester la logique d'une ligne de code, d'une méthode ou d'une fonction de manière isolée au reste du programme.

UI : User Interface – l'aspect visuel du site web, conception de l'interface utilisateur.

User story : Élément de la méthodologie AGILE qui comprend une description simple et compréhensible d'une exigence d'un utilisateur final, l'action effectuée et le résultat attendu.

User technique : Élément de la méthodologie AGILE qui représente une description du travail du développeur sans être totalement liée à l'utilisateur directement. *Par exemple : modélisation de la base de données.*