|  |
| --- |
| Ian BIALO  01/06/2018 |

|  |
| --- |
| Demathieu Bard |
| Notice d’utilisation du site de sponsoring |
| Comprendre le fonctionnement du Front Office et Back Office de l’application Web. |



Avant toute chose….

Ce document a été réalisé dans le but d’informer toutes les personnes voulant utiliser l’application web mais aussi toutes les personnes voulant la modifier ou l’améliorer. Toutes les indications concernant les différentes pages ainsi que toutes les fonctionnalités y sont inscrites selon les différents cas de figures possibles. Dans le cas où il subsisterait encore divers problèmes tels que des redirections vers des pages inexistantes, des erreurs d’application ou un non fonctionnement de certaines fonctionnalités, veuillez contacter l’administrateur du site.

L’application web de demande de partenariat, de sponsoring ou encore de mécénat a été réalisé dans le but de répondre aux besoins du département de la communication de l’entreprise Demathieu Bard et plus particulièrement l’automatisation des traitements des données liées aux demandes et l’amélioration du suivi des projets liés à ceux-ci. L’application web peut être distinguée en deux parties à savoir une partie comprenant un formulaire permettant la création d’un projet (le FrontOffice) et une partie de suivi des projets (le BackOffice).

Une version originale de l’application est disponible à l’adresse suivante : <https://github.com/iancdp9/DBProject> . Les consignes d’installation se situent dans le fichier README disponible dans le dépôt GitHub.

Sommaire

Table des matières

[I. Installation de l’application 2](#_Toc515892388)

[II. Spécificités techniques et structure du projet 2](#_Toc515892389)

[1) Outils de développement 2](#_Toc515892390)

[2) Structure du projet 3](#_Toc515892391)

[A. Structure à la racine 3](#_Toc515892392)

[B. Structure au dossier source 5](#_Toc515892393)

[3) Fonctionnement de l’application 6](#_Toc515892394)

[A. Fonctionnement vis-à-vis d’une méthode GET 7](#_Toc515892395)

[B. Fonctionnement vis-à-vis d’une méthode POST 7](#_Toc515892396)

[C. Complément de fonctionnement 8](#_Toc515892397)

[III. Disposition du site et fonctionnalités associées 8](#_Toc515892398)

[1) Le FrontOffice 9](#_Toc515892399)

[A. Le formulaire 9](#_Toc515892400)

[B. L’après-formulaire 15](#_Toc515892401)

[2) Le BackOffice 17](#_Toc515892402)

# Installation de l’application

# Spécificités techniques et structure du projet

Si vous n’êtes pas à l’aise avec les notions de programmation web ou que vous souhaitez seulement vous intéresser aux fonctionnalités ainsi qu’à la disposition du site, veuillez vous référer au point (III.) de ce guide.

## Outils de développement

L’application web a été développé sous PHP 7.2.3 avec l’utilisation de l’IDE Eclipse Oxygen 3, un serveur apache 2.4.29 et un système de gestion de base de données MySQL 10.1.31 (MariaDB).

En complément, divers outils supplémentaires ont été ajoutés pour répondre aux divers besoins de l’application. Parmi ces outils on retrouve :

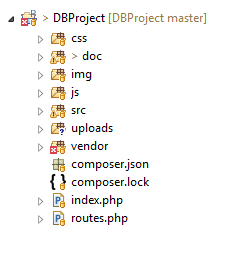
* Eloquent : micro-Framework issu du framework PHP Laravel utilisé pour interagir avec la base de données avec l’utilisation d’un modèle d’architecture ActiveRecord.
* Slim (v2) : micro-Framework utilisé en tant que Dispatcher ou Routeur d’URLs au sein du projet. Concrètement, il permet de lier certaines routes à des méthodes de classe pour la génération personnalisée de code HTML.
* PHPMailer : librairie alternative à la fonction d’envoi de mail de base de PHP.
* Materialize : framawork CSS utilisé pour la mise en place exclusive des éléments de la partie back-office.

L’application a dû être par la suite réadapté vis-à-vis de la version de PHP antérieure du serveur devant accueillir l’application. Il s’agissait donc de réintégrer les bonnes versions des outils supplémentaires.

## Structure du projet

### Structure à la racine

La structure de l’application web a été pensée de manière à suivre une l’architecture logicielle MVC (Modèle-Vue-Contrôleur). On retrouve en plus tous les fichiers nécessaires au bon développement de l’application dans les dossiers dédiés.



Structure générale de l’application web

A la racine du projet on retrouve :

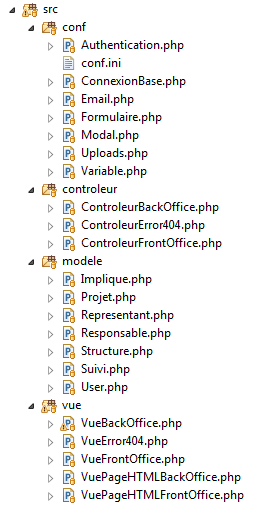
* Le dossier css comprenant toutes les feuilles de style en cascade (plus communément appelés fichiers CSS) utilisés pour la mise en place des éléments sur toutes les pages de l’application.
* Le dossier doc comprenant toutes la documentation liée à l’application, dont ce guide d’utilisation mais aussi les différents schémas d’utilisation.
* Le dossier img comprenant toutes les fichiers au format .jpeg et .png utilisées sur les différentes pages.
* Le dossier js comprenant tout les fichiers JavaScript utilisées en complément de toutes les fonctionnalités qu’apporte PHP utilisées sur les pages web (côté client).
* Le dossier uploads qui contient tous les fichiers uploadés par les utilisateurs de l’application (il est donc initialement vide).
* Le dossier vendor comprenant tous les outils supplémentaires au bon développement de l’application.
* Le fichier index.php qui permet d’instancier tous les éléments de l’application.
* Le fichier routes.php qui définit toutes les routes de l’application (ce fichier utilise Slim vu précédemment).
* Tous les fichiers de configuration.

Il est à noter que la génération du vendor et donc la génération des modules complémentaires à l’application s’effectue à l’aide de composer qui permet donc de déclarer et d’installer des bibliothèques dans le projet. Toutes les informations sont recueillies dans le fichier composer.json et la génération s’effectue à l’aide de la commande :

**composer install**

Dans le cas où une version d’un module ne serait plus utilisable à cause d’une mauvaise version ou d’une mauvaise interaction des différent modules (ce qui n’est pas censé arriver), il suffit juste de modifier les paramètres du fichier composer.json.

### Structure au dossier source



Structure du dossier src de l’application web

Au niveau du dossier de source de l’application web, on retrouve les éléments regroupés selon l’architecture MVC vu précédemment. Etant donné que l’application web compte une partie liée au FrontOffice et une partie liée au BackOffice, il était nécessaire de faire une distinction de ceux-ci dans les fichiers de l’application. Il y a donc :

* Le dossier conf comprenant tout les fichiers PHP avec des méthodes statiques utilisées dans l’application selon les disponibilités. Ces fichiers sont :
* Authentication.php : fichier qui permet à l’utilisateur de se connecter, de créer un compte, de se déconnecter de la partie BackOffice de l’application.
* ConnexionBase.php : fichier qui permet d’établir la connexion à la base de données MySQL (ce fichier utilise Eloquent vu précédemment).
* Email.php : fichier permettant la gestion d’envoi de mails au travers de l’application.
* Formulaire.php : fichier permettant la gestion des formulaires lorsqu’un utilisateur décide de créer un nouveau projet au travers du formulaire du FrontOffice de l’application.
* Modal.php : fichier utilisé dans les vues pour éviter la réutilisation la réécriture de code inutile. Il permet la génération de boîte de confirmation.
* Uploads.php : fichier permettant la gestion d’enregistrement et de suppression de fichier au sein de l’application.
* Variable.php : fichier contenant toutes **les variables pratiques et modifiables** de l’application. Cela signifie que lors de la configuration de l’application sur un serveur web, ce fichier doit être utilisé pour pouvoir modifier le fonctionnement de l’application selon les nécessités.

En plus de ces fichiers, on retrouve un fichier de configuration conf.ini **inexistant** lors de la récupération du projet du dépôt GitHub. Il contient toutes les informations de connexion à la base de données et est utilisé dans le fichier ConnexionBase.php.

* Le dossier controleur qui contient distinctivement les fichiers de contrôle de la partie FrontOffice et BackOffice de l’application mais aussi le fichier de contrôle pour gérer le cas des erreurs 404 (ressource demandée inexistante) venant de l’application web.
* Le dossier modele qui contient les modèles de l’application. Un modèle correspond à une table dans la base de données et ce sont ces modèles qui sont appelés dès lors que la manipulation des données est nécessaire. Ces fichiers utilisent Eloquent vu précédemment.
* Le dossier vu qui contient toutes les vues de l’application. En plus des vues liées au BackOffice, FrontOffice et la Vue liée à l’erreur 404, on retrouve les fichiers VuePageHTML qui correspondent à des fichiers ayant des méthodes statiques permettant de générer selon les cas de figures différentes balises <head> ou encore <footer> selon les nécessitées.

## Fonctionnement de l’application

L’application Web fonctionne pour chaque cas de figure de la même manière. On différencie l’exécution selon les types de requêtes effectuées c’est-à-dire les méthode GET et POST. Il est à noter que l’application différencie les URLs selon les différentes requêtes, ce qui signifie qu’une URL peut être identique du moment que les méthodes sont différentes.

### Fonctionnement vis-à-vis d’une méthode GET

Lorsqu’une méthode GET est envoyée au serveur, il réagit de cette manière :

* Premièrement, il vérifie que l’URL est une URL qui est enregistrée. L’application accède au fichier index.php pour ensuite accéder au fichier routes.php et chercher la route qui lui a été indiquée.
* Si la route a été trouvée, il accède à la fonction anonyme liée à la route et l’exécute.
* Si la route n’a pas été trouvée, il accède à une route par défaut qui lance la fonction notFound de Slim et qui accède et exécute la fonction anonyme liée.
* Ensuite l’application exécute tout ce qui se trouve dans la fonction anonyme.
* Il va premièrement chercher la première partie du code HTML souhaitée selon les différents cas de figures.
* Il va ensuite accéder au controleur spécifique.
* Il va finir par chercher la dernière partie du code HTML souhaitée.
* Pendant l’appel du controleur spécifique, l’application prépare les différentes données qui seront réutilisées dans l’application. Dans le cas du BackOffice, l’application va également effectuer des tests de sécurité pour contrôler les personnes qui accèdent au controleur : cela est fait par exemple avec la vérification de la connexion de l’utilisateur à l’application. Si tout s’est bien déroulé, l’application accède à la vue qui lui est associée.
* Pendant l’appel de la vue, un code HTML est généré et retourné dans le controleur. Cette vue fait appel aux données de la base de données dans le cas du BackOffice. Dans tous les cas, les vues sont générées dynamiquement.
* Finalement, le controleur ayant reçu la vue il peut l’afficher pour constituer la partie centrale du code HTML.

De cette manière, le code HTML complet est généré par l’application et est renvoyé au client pour l’afficher dans son navigateur.

### Fonctionnement vis-à-vis d’une méthode POST

Lorsqu’une méthode POST est envoyée au serveur, elle réagit d’une manière similaire lorsqu’une méthode GET est envoyée à la différence que l’application n’affiche rien et que aucune vue n’est sollicitée.

* Premièrement, il vérifie que l’URL est une URL qui est enregistrée. L’application accède au fichier index.php pour ensuite accéder au fichier routes.php et chercher la route qui lui a été indiquée.
* Si la route a été trouvée, il accède à la fonction anonyme liée à la route et l’exécute.
* Si la route n’a pas été trouvée, il accède à une route par défaut qui lance la fonction notFound de Slim et qui accède et exécute la fonction anonyme liée.
* Ensuite l’application exécute tout ce qui se trouve dans la fonction anonyme. Il exécute dans tout les cas un appel à une méthode du controleur spécifique.
* L’application exécute le code de la fonction appelée pour finalement faire appel à une méthode GET. Cela permet donc de rediriger l’application vers une méthode qui permettra d’effectuer un affichage pour informer l’utilisateur de la réussite ou non de l’application après l’action de la méthode POST. La redirection s’effectue à l’aide d’une méthode vu au point (II.3) C.) ci-dessous.

### Complément de fonctionnement

Il est important de préciser que Slim permet de générer des URLs et d’effectuer des redirections au sein de l’application :

* urlFor : méthode de Slim permettant de retrouver l’URL demandée. Il prend en paramètre un nom qui est le nom donné à l’URL dans le fichier route.php à l’aide de la méthode name().
* redirect : méthode de Slim permettant de rediriger l’application vers une méthode GET. Il prend en paramètre une URL qui peut être généré avec la méthode urlFor vu précédemment.

Ces méthodes sont constamment utilisées dans l’application et permettent un accès facile aux URLs de l’application. Lors de l’exécution d’une méthode POST, il s’agit de la méthode redirect qui est appelée pour rediriger l’application vers une méthode GET.

# Disposition du site et fonctionnalités associées

L’application web de demande de partenariat, de sponsoring ou encore de mécénat est une application qui été réalisé dans le but de répondre aux besoins du département de la communication de l’entreprise Demathieu Bard, cela signifie donc que toutes les fonctionnalités et toutes les pages du site ont été conçus de manière à répondre à leurs besoins. Si une page ou une fonctionnalité ne vous semble pas pertinent voir même inutile, contactez le service de communication de l’entreprise.

L’objectif de l’application est donc de pouvoir permettre à des utilisateurs externes à l’entreprise de pouvoir remplir un formulaire concernant une demande de partenariat, de sponsoring ou encore de mécénat et de pouvoir traiter tous ces formulaires en parallèle. L’idée de l’application est donc de différencier le site en deux parties (vu dans la point (II.) :

* Une partie contenant le formulaire : le FrontOffice.
* Une partie contenant une interface de gestion des formulaires et de suivi : le BackOffice.

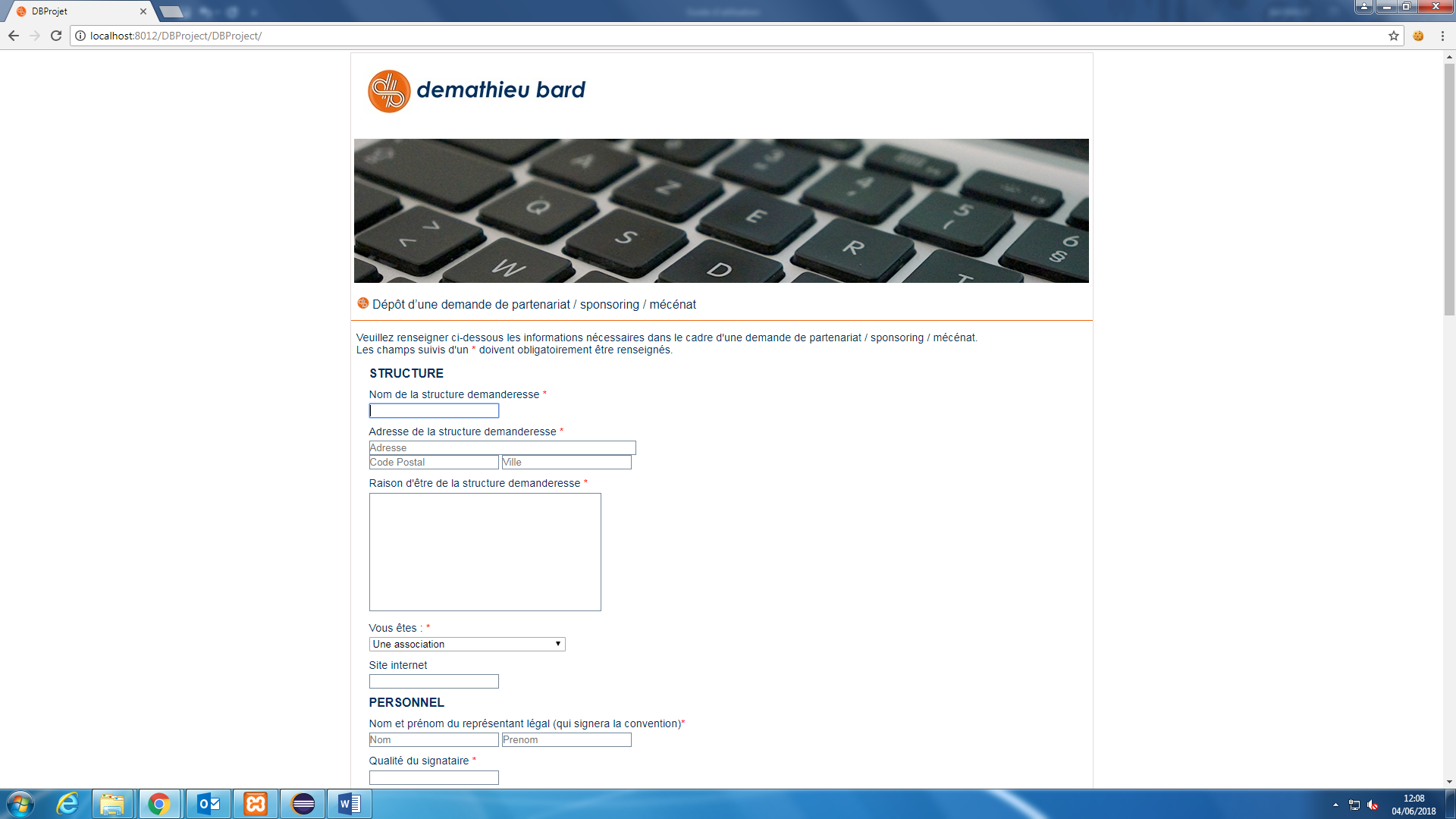
Pour la suite du guide, nous allons imaginer que l’application web est mis en place et qu’il est accessible à l’adresse [**https://www.sitedb.com/partenariat/**](https://www.sitedb.com/partenariat/). Bien évidemment le chemin redirige vers une page inexistante puisqu’il s’agit d’une adresse imaginaire qui sert juste d’exemple et qui n’est pas utilisé par l’entreprise.

## Le FrontOffice

Le FrontOffice est, comme il a été dit précédemment, la partie accessible à toutes les personnes souhaitant remplir un formulaire de demande de partenariat, de sponsoring ou encore de mécénat. Il s’agit donc d’une partie publique qui est rattachée au site existant de l’entreprise c’est pour l’application tend à respecter la charte graphique du site déjà existant.

Il s’agit de la racine du projet, c’est pourquoi il est accessible en indiquant le chemin seul à l’adresse <https://www.sitedb.com/partenariat/>.

### Le formulaire



Page du formulaire du FrontOffice (coupée)

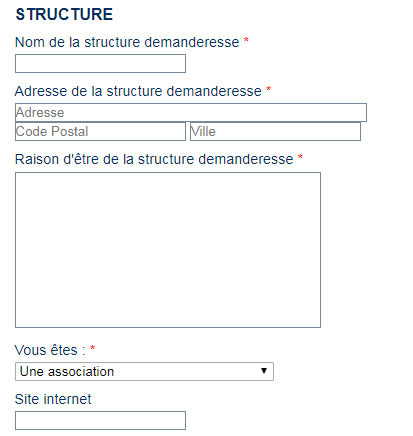
La page du formulaire se distingue en deux parties : la partie supérieure qui, en plus du logo et d’une image permettant d’aérer la page, donne des indications pour les personnes souhaitant remplir le formulaire et la partie inférieure qui contient le formulaire.

Le formulaire est structuré selon trois catégories :

* Structure : coordonnées de la structure concernée par la demande.
* Personnel : coordonnées des personnes concernées par la demande.
* Projet : descriptif du projet pour la demande

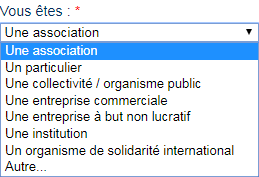
Chaque champ comprenant un petit texte décrivant ce qui doit être inséré. Dans le cas où un texte se finit par une étoile en rouge (\*), cela signifie que le champ doit être obligatoirement complété

#### Structure



Champs associés à la structure dans le formulaire

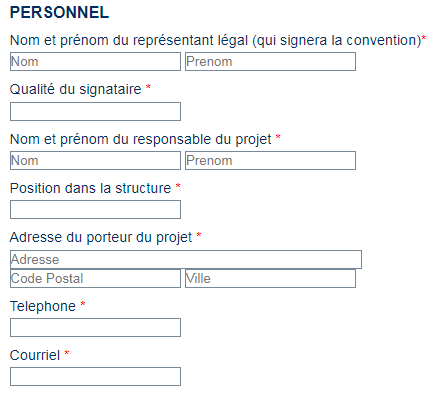
Dans la catégorie « structure », six champs obligatoires sont proposés : nom, adresse, code postal, ville et type de la structure demanderesse ainsi qu’un champ non-obligatoire qui est le site internet.



Champ associé au type de structure dans le formulaire

Tous les champs sont des champs à compléter au clavier sauf le champ « type » qui propose un choix de réponses possible. Néanmoins il est possible de choisir « Autre » comme type de structure pour faire apparaître un nouveau champ obligatoire qui demande à l’utilisateur de préciser le type de structure.

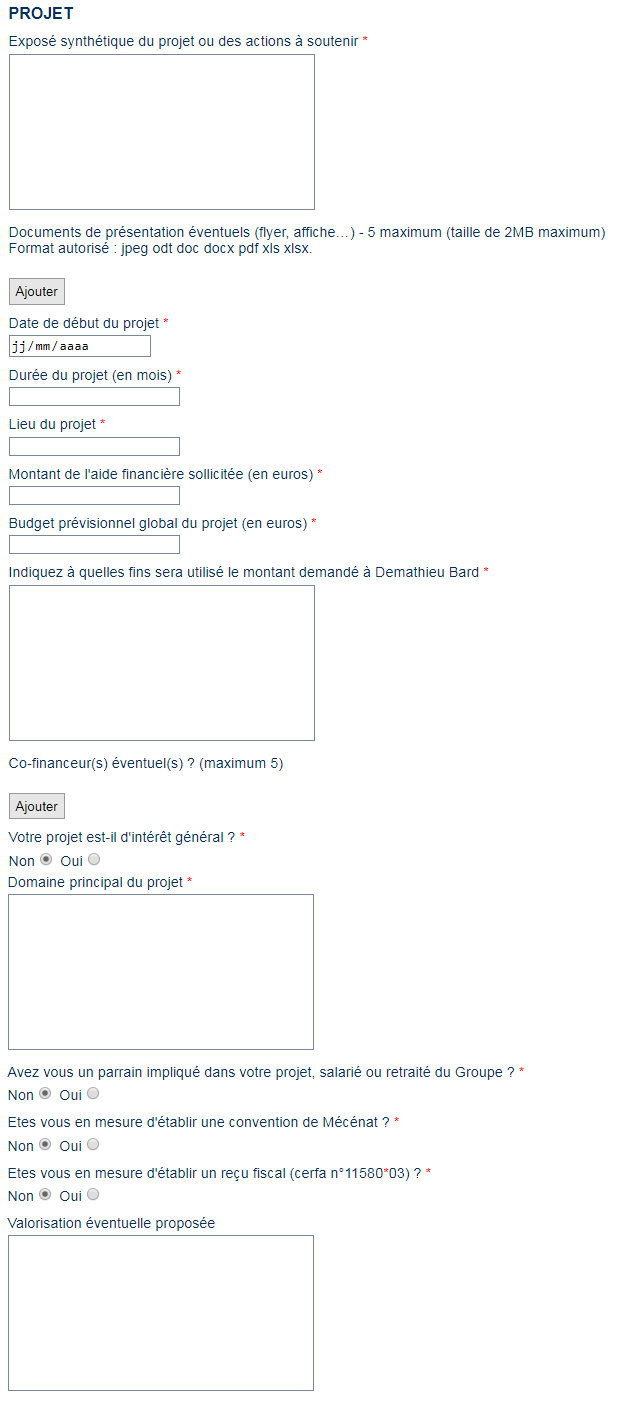
#### Personnel



Champs associés au personnel dans le formulaire

Dans la catégorie « Personnel », on retrouve trois champs obligatoires concernant le représentant légal : nom, prénom et qualité. On retrouve également huit champs obligatoires concernant le responsable du projet : nom, prénom, position, adresse, code postal, ville, téléphone et courriel. Aucun champ non-obligatoire n’est présent dans cette catégorie.

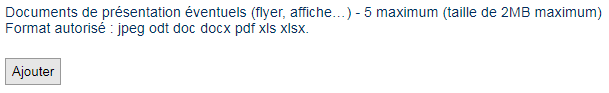
#### Projet

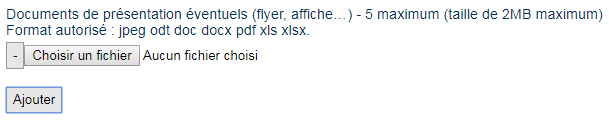


Champs associés au projet dans le formulaire

Dans la catégorie principale « Projet », on retrouve beaucoup plus de champs que dans les catégories précédentes. Huit champs obligatoires où il faut rentrer les informations au clavier sont comptabilisés dans cette catégorie : exposé synthétique, date de début de projet, durée du projet, lieu du projet, montant de l’aide, budget prévisionnel, les fins, et le domaine principal du projet. On retrouve également quatre champs obligatoires VRAI/FAUX concernant le type de projet (s’il est d’intérêt général), les parrains impliqués, la convention de Mécénat et le reçu fiscal (voir schéma ci-dessus pour plus de précision).

Concernant les champs non-obligatoires on retrouve un champ proposant d’ajouter une valorisation éventuelle au projet mais aussi deux boutons permettant d’ajouter des documents au projet et d’ajouter d’éventuels co-financeurs.



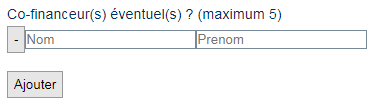
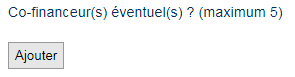


Champ associé à l’ajout de fichier au projet

Il est possible d’ajouter des fichiers au projet. Pour cela, il suffit d’appuyer sur le bouton « Ajouter » pour ensuite faire apparaître un champ obligatoire d’ajout de fichier. Il est possible d’ajouter des fichiers en respectant des conditions :

* Le nombre de fichier doit être au plus de cinq.
* La taille maximale de chaque fichier doit être de deux MB au plus.
* Le format doit être un des suivants : jpeg, odt, doc, docx, pdf, xls, xlsx.

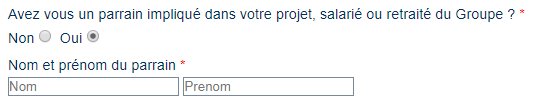
Pour supprimer un fichier, il suffit d’appuyer sur le bouton « - » à côté du fichier en question.



Champs associés à l’ajout de co-financeurs au projet

Il est possible d’ajouter dynamiquement des co-financeurs au projet. Pour cela, il suffit d’appuyer sur le bouton « Ajouter » pour ensuite faire apparaître un champ obligatoire d’ajout de co-financeurs. La seule contrainte imposée est de ne pas pouvoir insérer plus de 5 co-financeurs.





Champs associés à l’ajout d’un parrain au projet

Il est possible d’ajouter un parrain au projet. Pour cela, il suffit de mentionner le fait qu’un parrain est impliqué dans le projet en cochant l’option « Oui » et deux nouveaux champs obligatoires feront leurs apparitions. Si vous souhaitez faire disparaître ces champs, cocher l’option « Non ».

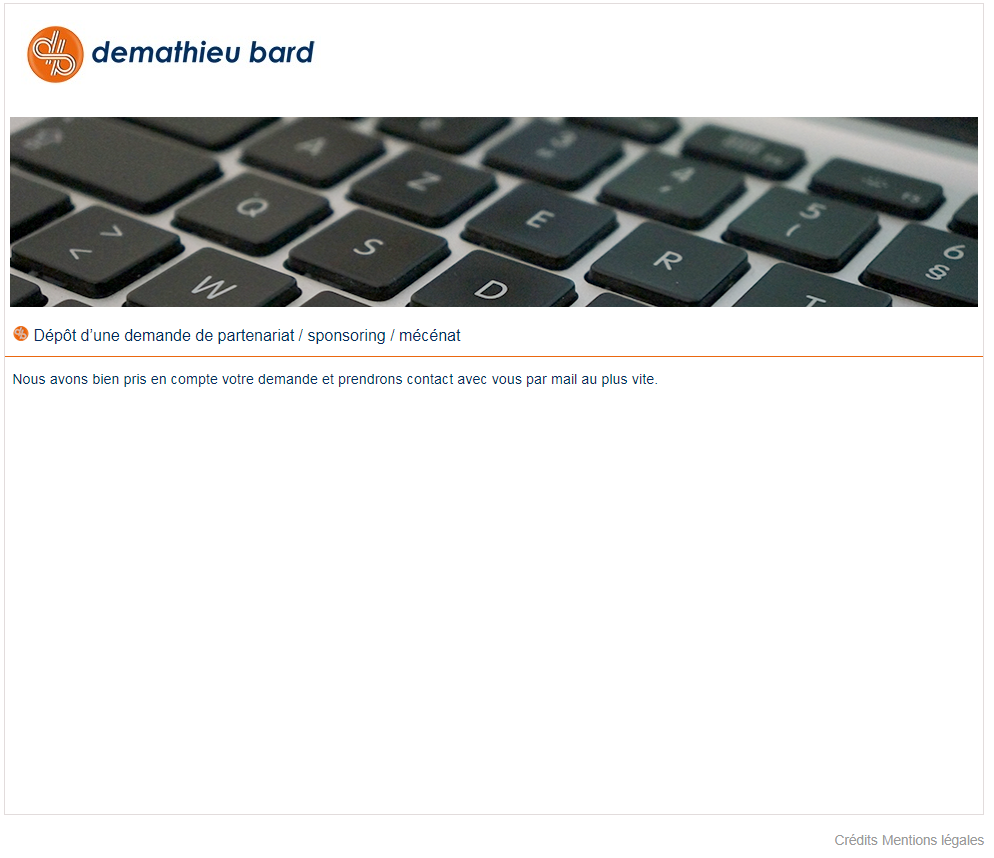


Lorsque le formulaire est complété dans son intégralité, pour le valider il suffit d’appuyer sur le bouton « Valider et envoyer votre demande » situé en bas de page. Si aucune redirection n’a été faite, cela signifie qu’un champ n’a pas été renseigné et la page web vous redirigera vers le champ incomplet.

### L’après-formulaire

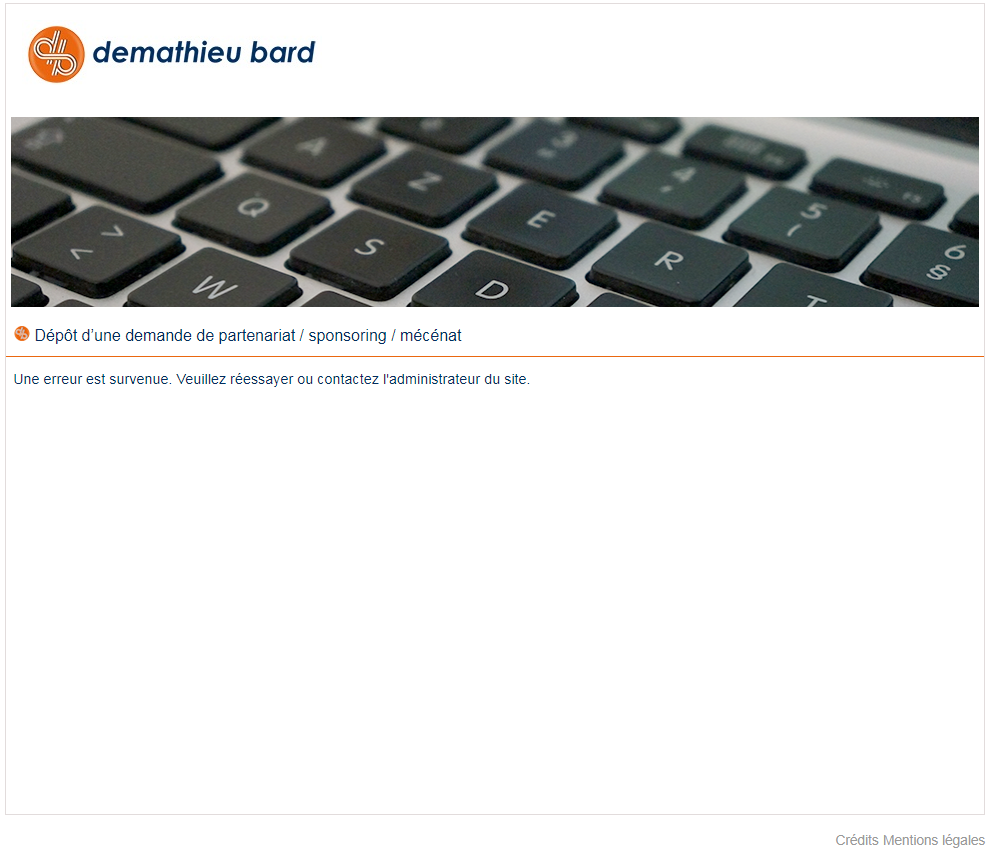
Dès lors que le formulaire a été complété et validé, l’application web se charge de récupérer toutes ces données et de les insérer dans la base de données mais aussi de stocker les fichiers envoyés par l’utilisateur :

* L’application récupère les informations concernant la structure, le responsable, la structure et les insères dans la base de données.
* L’application insère un nouveau suivi lié au projet et l’insère dans la base de données.
* L’application crée un nouveau projet, récupère les informations, le lie à la structure, au responsable et à la structure créée et insérée dans la base de données et l’insère à son tour dans la base de données.
* L’application crée les personnes impliquées (les co-financeurs et les parrains), les lient au projet et les insère dans la base de données.
* L’application récupère les fichiers envoyés par l’utilisateur, les analyse et les enregistre dans le dossier prévu à cet effet.
* L’application notifie les personnes chargées de traiter ces demandes au travers d’un mail.



Page du message de réussite du FrontOffice

Si tout se déroule comme il a été mentionné ci-dessus, alors l’application redirige l’utilisateur vers une nouvelle page où un message explique que l’enregistrement a été correctement effectué et que l’entreprise se chargera de contacter au plus vite l’utilisateur.



Page de message d’échec du FrontOffice

Dans le cas contraire, un message notifie l’utilisateur de l’échec avec la redirection de celui-ci vers une page dédiée.

## Le BackOffice