|  |  |
| --- | --- |
| **AFPA DWWM** | [|>Table des matières<|](#table_des_matieres) |

**Développement Web Fullstack**

**Par Philippe Giraud**

**Projet semaine du 25 Avril 2022**

**Sujet :**

Ce travail doit vous permettre de constituer des livrables qui pourront prendre place dans votre dossier projet.

Sur les bases de votre projet fil rouge vous devez :

1. Définir les différents types de users de votre projet (personas) | 2022-04-24
2. Définir les user stories de l’ensemble de votre projet, de la façon la plus exhaustive possible (formalisme Scrum) | 2022-04-25
3. Réaliser un ou plusieurs diagrammes des use case (formalisme UML) | 2022-04-25
4. Concevoir le MCD répondant aux user stories identifiées (JMerise) | 2022-04-26
5. Dans Symfony :
6. Mettre en place les entités principales | début 2022-04-26
7. Intégrer le frontend du front office dans un layout |
8. Définir les routes des pages principales du front office |
9. Intégrer ces pages en statique |

**Livraison :**

La livraison sera faite par l’intermédiaire d’un dépôt GitHub sur votre compte habituel (projet Symfony avec un dossier Conception).| dépôt crée le 2022-04-26

Nom du dépôt | rec01-fanzine-de-breizh

**Attention :** vous devez créer un nouveau dépôt et ne pas travailler dans les dépôts précédents.| Ok

|  |  |
| --- | --- |
| **Table des matières** |  |

|  |  |
| --- | --- |
| [Table des matières](#table_des_matieres) | 2 |
| [Présentation du projet](#presentation_du_projet) | 3 |
| [Personas (visiteurs et visiteurs identifiés)](#personas_visiteurs) | 4 |
| [Personas (équipe gestionnaire du site)](#personas_back_end) | 5 |
| [User Story (visiteurs)](#user_stories_visiteurs) | 6 |
| [User Story (équipe admin)](#user_stories_back_end) | 7 |
| [Use Case (front Office)](#use_case_front_end) | 8 |
| [Use Case (back end)](#use_case_back_end) | 9 |
| [Drafts (réflexion sur les champs que devraient contenir les entités)](#drafts) | 10 |
| [Méthode Merise](#methode_merise) | 11 |
| [Drafts (base de données : villes)](#drafts_bases_de_données_01) | 12 |
| [Drafts (base de données : départements)](#drafts_bases_de_données_02) | 13 |
| [MCD (Modèle conceptuel des données)](#modele_conceptuel_des_donnees) | 14 |
| [MLD (Modèle logique des données)](#modele_logique_des_donnees) | 15 |
| [Requêtes issues de JMERISE (Modèle physique des données)](#requetes_issues_de_merise) | 17 |
| [Problématiques rencontrées JMERISE](#problematiques_jmerise) | 18 |
| [Livraison du projet : repository GIT](#livraison_du_projet) | 19 |
| [Rappel : ce qu’est une application MVC](#rappel_application_mvc) | 20 |
| [Symfony : Mise en place des entités](#symfony_mise_en_place_des_entites) (démarrage du projet) | 21 |
| [Symfony : Mise en place des entités (création du premier controller)](#symfony_creation_premier_controller) | 22 |
| [Symfony : intégration du frontend](#symfony_integration_frontend) | 23 |
| [Symfony : extrait de la charte graphique](#symfony_charte_graphique) | 24 |
| [Symfony : intégration du template dans le layout](#symfony_integration_template_dans_layout) | 25 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| [Arborescence de mon projet](#arborescence_du_projet) | xx |
| [Arborescence de mon projet | dossiers conception et config](#arborescence_du_projet_suite_01) | xx |
| [Arborescence de mon projet | dossiers migrations et public](#arborescence_du_projet_suite_02) | xx |
| [Arborescence de mon projet | dossier src](#arborescence_du_projet_suite_03) | xx |
| [Arborescence de mon projet | dossier templates](#arborescence_du_projet_suite_04) | xx |
| [Arborescence de mon projet | résumé](#arborescence_du_projet_resume) | xx |
| [Astuces](#astuces) | xx |
| *Outils utilisés pour rédiger ce document :*  *Figma*  *GIMP*  *JMerise*  *Modelio Open Source*  *Notepad++*  *Outil capture d’écran*  *Paint*  *Paint 3D*  *Word* | |

|  |  |
| --- | --- |
| **Présentation du projet** |  |

Mon fil rouge est un blog où je décris les balades que je fais lorsque je suis en Bretagne, et en Cornouaille tout particulièrement. Je m’inspire donc en partie du travail déjà réalisé pour faire le TP. Je vais cependant prendre deux templates Bootstrap pour habiller le blog. Un pour le front-end et un pour le back-end. Ne sachant pas ce que me réserve ma période en entreprise, je vais considérer que ce projet pourrait être présenté au jury.

Au fil du temps, le projet a évolué. La réalisation des premiers personas que j’aurais dû rendre lors de la compétence 2, m’avait permis de constater que le blog devait être plus interactif et permettre aux visiteurs de déposer un commentaire. La poursuite des apprentissages et notamment Symfony me conforte dans l’idée que des évolutions doivent encore se faire jour selon les compétences à couvrir.

Le projet n’étant achevé sur le papier, ce TP me donne l’occasion et me force à penser à tous les mécanismes fonctionnels et de sécurité qui doivent être présents pour un fonctionnement fluide du blog.

*Ci-dessous, description à mettre de mon blog initial et du template vers lequel je vais migrer.*

|  |  |
| --- | --- |
| **Personas** |  |

**Visiteurs**

J’ai imaginé trois profils de visiteurs qui pourraient se rendre sur le site pour lire les articles et poster un commentaire. Ce sont les visiteurs belges qui m’ont convaincu de modifier le blog pour y rajouter la possibilité de poster un commentaire.

|  |  |
| --- | --- |
| **La famille Weiler**  Habitent à Hartmannswiller en Alsace.  Engagés dans la vie locale, ils apportent leur aide aux associations qui œuvrent pour les plus démunis. |  |

|  |  |
| --- | --- |
| **Hortense Le Cleac’h**  Habitent à Urrugne dans le Pays Basque.  Ne voit pas assez sa famille qui habite en Bretagne.  Aime s’adonner à la contemplation. |  |

|  |  |
| --- | --- |
| **Marieke et Nikolaas**  Habitent à Amsterdam aux Pays-Bas.  Ils parcourent avec leur camping-car la Bretagne tous les étés en compagnie de leurs petits-enfants.  Ils attendent beaucoup du site y compris s’il y a à proximité un parking pour les camping-cars. |  |

|  |  |
| --- | --- |
| **Personas** |  |

**Equipe back-end**

J’ai imaginé trois profils de personnes qui seraient amenées à gérer le site. Ce sont les cours sur Symfony qui ont précisé chez moi la formalisation de leurs profils.

Dans ma pensée, je suis le commanditaire, le client. L’équipe qui travaille sur mon projet est composée de trois personnes. Il y a un développeur, Mathieu, qui réalise les évolutions souhaitées, une rédactrice des contenus, Annabelle, qui est également modératrice et enfin, Natasha, qui est l’admin du site et qui tient également le rôle de chef de projet.

|  |  |
| --- | --- |
| **Mathieu Sainte-Rose**  Rêve d’un tour du monde à la voile.  Souhaite retourner à Saint Barth pour y ouvrir sa propre agence web et ainsi être plus proche de sa famille.  Rôle : Ne possède pas de rôle dans le CRUD du contenu du blog. |  |

|  |  |
| --- | --- |
| **Annabelle Liu**  Savoir que la planète se meurt à petit feu lui cause beaucoup de peine.  Engagée à plein temps, voudrait pouvoir accueillir un enfant.  Rôle : Rédactrice de publications et modératrice des commentaires postés par les visiteurs. |  |

|  |  |
| --- | --- |
| **Natasha Pontes**  Hyperactive, trouve qu’elle ne fait pas assez de sport alors qu’elle se dépense beaucoup.  Constamment à la recherche de solutions novatrices pour le client.  Frustrée de ne pas avoir ses enfants avec elle.  Rôle : Admin et chef de projet. |  |

|  |  |
| --- | --- |
| **User Stories** |  |

Plutôt que de faire un seul bloc ‘user story’, j’ai privilégié de faire deux vues, une pour les visiteurs du blog et une autre pour les actions qui pourront être effectuées par l’équipe en charge de l’administration du site. Ainsi la lisibilité sera plus grande. Aux deux tableaux, il y aura également deux vues graphiques associées qui permettront une vue d’ensemble.

**User story -> Visiteurs**

Définitions :

Visiteur : Personne qui visite le site web qui n’est pas identifiée.

Visiteur identifié : Personne qui visite le site web, qui s’est connectée avec son login et mot de passe.

J’ai imaginé qu’il pourra y avoir seulement ces deux rôles dans les visiteurs. Leurs actions pourront alors être différentes.

Tag : Mot-clé permettant d'identifier un élément ou un document.

Visiteur non identifié

|  |  |  |  |
| --- | --- | --- | --- |
|  | En tant que | Je veux | Pour |
| 1 | Visiteur | M’inscrire | Commenter un article |
| 2 | Visiteur | M’inscrire | Modifier mon commentaire |
| 3 | Visiteur | M’inscrire | Supprimer mon commentaire |
| 4 | Visiteur | M’inscrire | Publier des articles |
| 5 | Visiteur | Faire des recherches par catégories | Trouver des articles précis |
| 6 | Visiteur | Faire des recherches par auteurs | Trouver des articles précis |
| 7 | Visiteur | Faire des recherches par ‘tags’ | Trouver des articles précis |
| 8 | Visiteur | Faire des recherches par villes | Trouver des articles précis |
| 9 | Visiteur | Faire des recherches par départements | Trouver des articles précis |
| 10 | Visiteur | Choisir un article parmi une liste des articles les plus récents | Lire des articles avec de toutes nouvelles balades |
| 11 | Visiteur | Choisir un article parmi une liste des articles les plus anciens | Retrouver une balade qui m’avait intéressé |
| 12 | Visiteur | Lire des articles abrégés après les recherches | Choisir celui qui m’intéresse |
| 13 | Visiteur | Lire un article complet | M’informer |
| 14 | Visiteur | Pouvoir lire les commentaires d’un article | Me faire une opinion sur les avis publiés au sujet de la balade |
| 15 | Visiteur | Envoyer un message à l’équipe du site | Prendre contact |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Visiteur identifié

Les visiteurs identifiés héritent des droits des visiteurs non identifiés. Toutes les actions ci-dessus peuvent être réalisées par les visiteurs identifiés.

|  |  |  |  |
| --- | --- | --- | --- |
|  | En tant que | Je veux | Pour |
| 1 | Visiteur identifié | M’authentifier | Publier un article |
| 2 | Visiteur identifié | M’authentifier | Publier un commentaire |
| 3 | Visiteur identifié | M’authentifier | Supprimer mon commentaire |
| 4 | Visiteur identifié | M’authentifier | Modifier mon commentaire |
| 5 | Visiteur identifié | Publier un article | Informer les internautes |
| 6 | Visiteur identifié | Commenter un article | Donner mon avis sur une balade |
| 7 | Visiteur identifié | Supprimer mon commentaire | Le commentaire ne correspondait pas à l’article |
| 8 | Visiteur identifié | Modifier mon commentaire | Supprimer les fautes d’orthographe |
| 9 | Visiteur identifié | M’authentifier | Me déconnecter |
| 10 | Visiteur identifié | Me déconnecter | Ne plus être visible |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Je n’ai pas indiqué qu’un visiteur identifié, lorsqu’il publiait un article devait obligatoirement :

* Choisir une ville, dans un département (implicite), choisir une catégorie parmi une liste déroulante.

et sans que cela soit obligatoire, indiquer un ou plusieurs tags

|  |  |
| --- | --- |
| **User Stories** |  |

L’équipe back-end est composée de trois personnes.

Mathieu, le développeur Web qui ne possède pas de rôle dans les actions de gestion du blog, si ce n’est pendant les vacances d’une des deux autres personnes ; Annabelle qui tient les rôles de modératrice et de manager content et Natasha qui est l’admin du site et qui est aussi UX et UI Director.

**User story -> Back-End**

Equipe back-end

|  |  |  |  |
| --- | --- | --- | --- |
|  | En tant que | Je veux | Pour |
| 1 | Modératrice | M’authentifier | Publier des articles par ‘Tags’ |
| 2 | Modératrice | M’authentifier | Publier des articles par Catégories |
| 3 | Modératrice | M’authentifier | Publier des articles par Villes |
| 4 | Modératrice | M’authentifier | Publier des articles par Départements |
| 5 | Modératrice | M’authentifier | Gérer les articles |
| 6 | Modératrice | M’authentifier | Gérer les ‘Tags’ |
| 7 | Modératrice | M’authentifier | Gérer les Catégories |
| 8 | Modératrice | M’authentifier | Gérer les Villes |
| 9 | Modératrice | M’authentifier | Gérer les Départements |
| 10 | Modératrice | Publier des articles par ‘Tags’ | Qu’ils soient retrouvés facilement |
| 11 | Modératrice | Publier des articles par Catégories | Regrouper les articles de même nature |
| 12 | Modératrice | Publier des articles par Villes | Permettre des recherches plus ciblées |
| 13 | Modératrice | Publier des articles par Départements | Permettre des recherches plus ciblées |
| 14 | Modératrice | Gérer les articles | Veiller à ce qu’ils ne contiennent pas de propos injurieux |
| 15 | Modératrice | Modérer les commentaires | Veiller à ce qu’ils ne contiennent pas de propos offensants |
| 16 | Modératrice | M’authentifier | Me déconnecter |
| 17 | Modératrice | Me déconnecter | Ne plus être visible |
| 18 | Admin | M’authentifier | Gérer les utilisateurs |
| 19 | Admin | Gérer les utilisateurs | Créer des utilisateurs |
| 20 | Admin | Gérer les utilisateurs | Donner des droits supplémentaires aux utilisateurs |
| 21 | Admin | Gérer les utilisateurs | Effacer un utilisateur qui aurait eu une conduite offensante |
| 22 | Admin | M’authentifier | Me déconnecter |
| 23 | Admin | Me déconnecter | Ne plus être visible |
| 24 | Admin | Prendre connaissance du mail envoyé par un visiteur | Le lire |
| 25 | Admin | Apporter une réponse au mail du visiteur | Le tenir informé |

|  |  |
| --- | --- |
| **Use Case** |  |

Définitions :

Include : La relation "Include" est une relation entre 2 instances de cas d'utilisation telle que la réalisation de l'un nécessite la réalisation de l'autre.

Extend : La relation extend est une relation entre 2 instances de cas d'utilisation telle que A extend B signifie que le comportement de B peut être complété par le comportement de A. La relation extend indique une possibilité, un complément possible.

**Use case -> Front Office**

|  |
| --- |
|  |

|  |  |
| --- | --- |
| **Use Case** |  |

Définitions :

Include : La relation "Include" est une relation entre 2 instances de cas d'utilisation telle que la réalisation de l'un nécessite la réalisation de l'autre.

Extend : La relation extend est une relation entre 2 instances de cas d'utilisation telle que A extend B signifie que le comportement de B peut être complété par le comportement de A. La relation extend indique une possibilité, un complément possible.

**Use case -> Back-End**

|  |
| --- |
|  |

|  |  |
| --- | --- |
| **Drafts** |  |

Les trois pages qui suivent sont mes réflexions avant la réalisation (pour mémo)

Entités :

Users | date de création | nom | prénom | email | authentifié | rôle |

Articles | date de publication | titre | post | image | tag | auteur | catégorie | ville |

Catégories | date de création | catégorie | description |

Villes | date de création |

Départements | date de création |

Connexions | user | connecté |

|  |  |
| --- | --- |
| **Méthode Merise** |  |

**Méthode Merise ; Qu’est-ce que c’est ?**

Merise est une méthode d'analyse, de conception et de gestion de projet informatique.

|  |
| --- |
|  |

*© Wikipédia*

|  |  |
| --- | --- |
| **Base de données** |  |

J’ai voulu concevoir un modèle qui permettrait d’associer aux articles publiés, des villes et des départements Les balades s’effectuant pour l’instant uniquement en France, sans qu’il y ait des fautes d’orthographes dans le nom des villes.

J’ai donc créé deux tables où j’ai inséré la totalité des villes et des départements français. Pour se faire je suis allé sur le site SQL.sh pour récupérer ces deux tables.

|  |  |
| --- | --- |
| **Tables issues de** | SQL |

Villes

|  |  |
| --- | --- |
|  |  |

Nota : il est vrai que le nombre d’informations présentes dans la table est très important, néanmoins pourrais-je peut-être apporter plus tard des modifs sur le blog afin de montrer où se trouvent la ville dans laquelle la balade a été effectué. Sous la forme d’une carte avec un localisation des coordonnées GPS.

Je laisse donc l’exhaustivité des infos des tables villes et départements en l’état.

|  |  |
| --- | --- |
| **Base de données** |  |

J’ai voulu concevoir un modèle qui permettrait d’associer aux articles publiés, des villes et des départements Les balades s’effectuant pour l’instant uniquement en France, sans qu’il y ait des fautes d’orthographes dans le nom des villes.

J’ai donc créé deux tables où j’ai inséré la totalité des villes et des départements français. Pour se faire je suis allé sur le site SQL.sh pour récupérer ces deux tables.

|  |  |
| --- | --- |
| **Tables issues de** | SQL |

Départements

|  |  |
| --- | --- |
|  |  |

Nota : il est vrai que le nombre d’informations présentes dans la table est très important, néanmoins pourrais-je peut-être apporter plus tard des modifs sur le blog afin de montrer où se trouvent la ville dans laquelle la balade a été effectué. Sous la forme d’une carte avec un localisation des coordonnées GPS.

Je laisse donc l’exhaustivité des infos des tables villes et départements en l’état.

|  |  |
| --- | --- |
| **MCD (modèle conceptuel des données)** |  |

|  |
| --- |
|  |

Pour exemple, j’ai indiqué quelques commentaires pour expliquer les cardinalités.

|  |  |
| --- | --- |
| **MLD (modèle logique des données)** |  |

|  |
| --- |
|  |

|  |  |
| --- | --- |
| **Requêtes issues de JMERISE (Modèle physique des données)** |  |

|  |  |
| --- | --- |
|  |  |

J’ai voulu tester l’import des requêtes SQL dans un base de données. Cela ne fonctionne pas !!

Les lignes :

,CONSTRAINT DEPARTEMENTS\_Idx INDEX (departement\_code,departement\_slug,departement\_nom\_soundex)

dans la création de la table ‘departements’

et la ligne :

,CONSTRAINT VILLES\_Idx INDEX (ville\_departement,ville\_slug,ville\_nom,ville\_nom\_simple,ville\_nom\_reel,ville\_nom\_soundex,ville\_nom\_metaphone,ville\_code\_postal,ville\_code\_commune,ville\_population\_2010,ville\_longitude\_deg,ville\_latitude\_deg)

dans la création de la table ‘villes’ possèdent toutes deux apparemment une erreur de syntaxe.

|  |
| --- |
| En enlevant ces deux lignes, l’importation fonctionne. Il y a cependant des warnings dans la création de chaque table, identiques à celui-ci-dessous. |
|  |

|  |  |
| --- | --- |
| **Requêtes issues de JMERISE (Modèle physique des données)** |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
| **Problématiques rencontrées JMERISE** |  |

**Explications :**

Dans la première version de ma conception MCD, j’avais imaginé que l’on pourrait créer une entité avec les messages reçus et sur le même enregistrement, laisser des champs qui seraient remplis lors d’une réponse ultérieure.

Je ne suis pas parvenu à réaliser ce que je voulais exactement.

|  |  |
| --- | --- |
| Capture d’écran d’une zone du MCD initial. |  |
| Capture d’écran de la table MESSAGES initiale dans laquelle les réponses apportées auraient été contenues dans le même enregistrement. |  |

Les problématiques étaient les suivantes :

* Il n’était pas possible d’apporter plusieurs réponses au visiteur et à son message, ce qui pourrait être le besoin.
* Les deux derniers champs de clefs étrangères faisaient référence à la même propriété ‘user\_ID’ qui se trouve dans l’entité USERS et ces deux champs ne pouvaient pas être NULL. L’ajout automatique du dernier champ de clef étrangère ‘user\_ID\_USERS’ était lié à la relation ‘REPONDRE’ qui associait l’entité ‘USERS’ à l’entité ‘MESSAGES’. Il aurait été impossible de le remplir convenablement car il était défini à ‘NOT NULL’. Comment aurait-on su qui avait rédigé la réponse au visiteur ?

*Nota : à ce stade du développement, les lignes exportées par JMERISE dans le modèle physique des données sont trop nombreuses. J’ai donc procédé à un ‘copier-coller’ pour compléter le script de création de la table ‘COMMENTER’ qui était tronqué.*

|  |  |
| --- | --- |
| **Livraison du projet : repository GIT** |  |

**Explications :**

Dans la première version de ma conception MCD, j’avais imaginé que l’on pourrait créer une entité avec les messages reçus et sur le même enregistrement, laisser des champs qui seraient remplis lors d’une réponse ultérieure.

Je ne suis pas parvenu à réaliser ce que je voulais exactement.

|  |  |
| --- | --- |
| Capture d’écran du repository. |  |
| Fichier readme.MD régulièrement mis à jour au fur et à mesure de l’avancement du projet. |  |

|  |  |
| --- | --- |
| **Rappel : ce qu’est une application MVC** |  |

**Définition :**

MVC (Model-View-Controller ou Modèle-Vue-Contrôleur) est un modèle dans la conception de logiciels. Il met l'accent sur la séparation entre la logique métier et l'affichage du logiciel. Cette « séparation des préoccupations » permet une meilleure répartition du travail et une maintenance améliorée.

Les 3 parties du modèle de conception de logiciel MVC peuvent être décrites comme suit :

1. Model (modèle) : gère les données et la logique métier.
2. View (vue) : gère la disposition et l'affichage.
3. Controller (contrôleur) : achemine les commandes des parties "model" et "view".

|  |
| --- |
| Représentation des interactions entre le modèle, la vue et le contrôleur dans le cas d'une application web.  Le motif MVC a été créé dans le but de mettre en œuvre des interfaces utilisateur. Certains détails sont alignés avec le langage Smalltalk, mais les grandes lignes peuvent s'appliquer à n'importe quel environnement. Le cycle action→mise à jour→affichage induit par ce patron est bien adapté aux applications web. De plus le patron impose la séparation des sujets, et les balises HTML sont ainsi confinées aux vues, ce qui améliore la maintenabilité de l'application. C'est le framework pour applications web Ruby on Rails qui a apporté un regain d'intérêt pour ce patron.  Ce patron est utilisé par de nombreux frameworks pour applications web tels que Ruby on Rails, Django, ASP.NET MVC, Spring, Struts ou Apache Tapestry.  Dans la mise en œuvre classique du patron MVC, la vue attend des modifications du modèle, puis modifie la présentation des éléments visuels correspondants. Cette mise en œuvre est appliquée pour les applications de bureau avec des framework comme Swing. Le protocole HTTP ne permet pas cette mise en œuvre pour les applications web. Pour ces dernières, lors d'une action de l'utilisateur, le contenu de la vue est recalculé puis envoyé au client. |
|  |

En quelques mots :

1. L’utilisateur envoie une requête au **CONTRÔLEUR**
2. Le **CONTRÔLEUR** envoie une demande de données au **MODELE**
3. Le **MODELE** renvoie les données au **CONTRÔLEUR**
4. Le **CONTRÔLEUR** remplit la **VUE** avec les données reçues
5. La **VUE** envoie in fine une réponse en HTML à l’utilisateur

|  |  |
| --- | --- |
| **Symfony : Mise en place des entités** |  |

**Démarrage du projet**

Dans les pages qui suivent, je vais décrire toutes les actions et leurs commandes associées que j’ai faites pour essayer de parvenir au résultat final attendu. La colonne de gauche indiquera l’action et celle de droite, les commandes tapées.

|  |  |
| --- | --- |
| **Action** | **Commandes** |

La toute première chose que j’avais faite, avant même la production de tout document de conception, fut le choix d’un nom de dossier et la création du projet SYMFONY dans ce dossier

|  |  |
| --- | --- |
| Je me suis positionné à la racine du dossier ‘www’ de Wampserver | symfony new rec01-fanzine-de-breizh --version=5.4 --webapp |

* Puis création des livrables dont j’ai détaillé la conception depuis le début de ce document

|  |  |
| --- | --- |
| Ensuite mise en place du dépôt dans mon GIT et commits réguliers au fur et à mesure de l’avancement de la rédaction du projet. REC01 c’est pour le numéro de la recette, il pourra y avoir plusieurs dossiers dans GIT (02, 03, etc…). | https://github.com/antoine-baptista/rec01-fanzine-de-breizh.git |

|  |  |
| --- | --- |
| Je lance le serveur SYMFONY | symfony server:start |

|  |
| --- |
|  |
| Des erreurs ont été trouvées dans la barre de débogage. C’est dû au fait qu’il n’existe encore aucune route. |

|  |  |
| --- | --- |
| **Symfony : Mise en place des entités** |  |

**Création du premier Controller**

|  |  |
| --- | --- |
| **Action** | **Commandes** |

|  |  |
| --- | --- |
| Création d’un controller avec le Maker. Je choisis de le nommer ‘Test’. Il me servira dans l’élaboration globale du blog. | php bin/console make:controller |

|  |
| --- |
|  |

* Lors de l’appel du controller, un problème est survenu à cause du Webpack, je procède donc à sa suppression dans le composer.json et je lance composer update dans la console.

|  |  |
| --- | --- |
| Maintenant, je peux faire appel au controller.  Les infos en rouge dans la barre de débogage ont disparu. |  |

|  |  |
| --- | --- |
| **Symfony : Intégration du frontend** |  |

**Intégration du frontend**

Je suis parti d’un template ‘philosophy’ que j’ai trouvé sur internet sur le site UseBotstrap.com et qui pourra remplacer mon blog initial. Il nécessite néanmoins beaucoup de modifications pour parvenir à respecter ma charte graphique.

Quelques captures d’écran de ce template :

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

**Informations sur le template**

Site où je l’ai téléchargé : <https://usebootstrap.com/>

Category:Blog & Magazine

Framework : Bootstrap 4.0.0

Publish date : June 21, 2018

Publisher : Colorlib

Views : 3,726

Downloads : 1,447

|  |  |
| --- | --- |
| **Symfony : Intégration du frontend** |  |

**Extrait de la charte graphique**

Ci-dessous, un extrait de la charte graphique du blog de mon projet fil rouge nommé : « Balades en Cornouaille »

PALETTE DES COULEURS

Couleurs utilisées

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Jaune-Vert**  #BAC700  C : 7  M : 0  Y : 100  K : 22  R : 186  G : 199  B : 0 | **Rose**  #D50050  C : 0  M : 100  Y : 62  K : 16  R : 213  G : 0  B : 80 | **Cyan**  #3DA6C1  C : 68  M : 14  Y : 0  K : 24  R : 61  G : 166  B : 193 | **Cyan-Bleu**  #0D2859  C : 85  M : 55  Y : 0  K : 65  R : 13  G : 40  B : 89 |

LES TYPOGRAPHIES DE LA CHARTE

|  |  |
| --- | --- |
| Les polices : | Pacifico |
|  | Source Serif Pro Semibold |
|  | Trebuchet MS |
|  | Bebas Neue |

J’appliquerai tout l’univers graphique de mon blog initial au template ‘philosophy’. Logotype, couleurs, typographie ainsi que l’habillage graphique.

|  |  |
| --- | --- |
| **Symfony : Intégration du frontend** |  |

**Intégration du template dans le layout**

Etapes :

* Intégration du layout dans le dossier \public\fanzine
* Copie de base.html.twig en fanzine.html.twig
* Intégration du layout :  
  Header : intégrer le Title, css, js\_top

Footer : intégrer le js\_bottom

Voici les modifs apportées au template :

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

J’ai repris les codes couleurs de mon site mais beaucoup d’autres modifications seront à apporter, notamment pour la couleur des titres et des textes des articles.

|  |  |
| --- | --- |
| **Symfony : Intégration du frontend (suite)** |  |

**Intégration du template dans le layout**

Etapes :

* Intégration du layout dans le dossier \public\fanzine
* Copie de base.html.twig en fanzine.html.twig

En modifiant au fur et à mesure le template pour lui apporter ma charte graphique, je me rends compte qu’il y a plusieurs types d’articles qui accompagnent le texte :

* Des articles photo
* Des articles vidéo
* Des articles audio

Je décide donc de modifier la table ‘articles’ en lui ajoutant un champ : ‘article\_type’ qui sera un VARCHAR(20) et de modifier le champ ‘article\_url\_media’ qui contiendra le lien vers l’image, le fichier audio ou vidéo. Le rédacteur devra choisir parmi une liste déroulante le type d’article qu’il souhaite créer. Cette liste déroulante contiendra trois items, nul besoin de créer une table supplémentaire, ce sera écrit en dur dans le code.

* Création d’un controller Posts avec les routes :

/ : home

/post/{id} : post\_view

|  |  |
| --- | --- |
| **Arborescence de mon projet** |  |

**Arborescence**

En fonction du temps que je vais pouvoir consacrer à l’écriture des pages documentant l’arborescence d’un projet Symfony, je décrirais ci-dessous ce que contient chaque dossier et où il se situe dans l’architecture MVC.

Il est important pour moi d’avoir une vision claire notamment avec des images des endroits où sont stockés physiquement les différents fichiers ainsi que leurs fonctions.

|  |  |
| --- | --- |
| Structure du dossier pour le volume BABOU  Le numéro de série du volume est 302C-24D3  C:.  | .env  | .gitignore  | arrow\_up.ico  | composer.json  | composer.lock  | directorylist.txt  | docker-compose.override.yml  | docker-compose.yml  | README.md  | symfony.lock  |  +---assets  |  +---bin  |  +---conception  |  +---config  | +---packages  |  | \---routes  |  +---migrations  |  +---public  | \---fanzine  | +---assets  | | +---images  | | \---media  | +---css  | | \---font-awesome  | | +---css  | | \---fonts  | +---fonts  |  | \---js  +---src  |  | +---Controller  |  | +---Entity  |  | \---Repository  |  +---templates  |  | +---philosophy  |  | +---post  | |  | \---test  |  +---translations  |  +---var  |  +---vendor |  |

|  |  |
| --- | --- |
| **Arborescence de mon projet | dossiers conception et config** |  |

**Arborescence**

Détails :

|  |  |
| --- | --- |
| Dossier : conception  Tous les documents relatifs à la conception du projet y sont rangés :   * MCD, MLD * Personas * Ressources : ce sont les fichiers sql des villes et département issus du web * Use\_Cases * User\_Stories : J’ai rédigé les user stories dans mon conducteur, je voulais en faire une version avec des ‘post-it’, je n’ai pas eu le temps à ce jour. * Divers fichiers, dont ce document. |  |

|  |  |
| --- | --- |
| Dossier : config |  |

|  |  |
| --- | --- |
| **Arborescence de mon projet | dossiers migration et public** |  |

**Arborescence**

Détails :

|  |  |
| --- | --- |
| Dossier : migrations |  |

|  |  |
| --- | --- |
| Dossier : public  C’est là où sont rangés l’habillage du site, le css, les ressources en images, etc.. |  |

|  |  |
| --- | --- |
| **Arborescence de mon projet | dossier src** |  |

**Arborescence**

Détails :

|  |  |
| --- | --- |
| Dossier : src |  |

|  |  |
| --- | --- |
| **Arborescence de mon projet | dossier templates** |  |

**Arborescence**

Détails :

|  |  |
| --- | --- |
| Dossier : templates |  |

|  |  |
| --- | --- |
| **Arborescence de mon projet | résumé** |  |

**Arborescence**

Détails :

|  |
| --- |
|  |

|  |  |
| --- | --- |
| **Astuces** |  |

**Ci-dessous quelques astuces**

|  |  |
| --- | --- |
| Exporter l’arborescence d’un dossier vers un fichier | tree C:\wamp64\www\rec01-fanzine-de-breizh /a /f > directorylist.txt |

|  |  |
| --- | --- |
| Si l’on s’est trompé en créant une entité, en retapant la même commande et à la condition que l’entité existe déjà, le maker vous demandera si vous souhaitez écraser l’entité existante ou la mettre à jour. | php bin/console make:entity Category |

|  |  |
| --- | --- |
| Commande pour lancer MailDev. | maildev --hide-extensions STARTTLS |

|  |  |
| --- | --- |
| Commande pour lancer ‘messenger’ dans une console. | php bin/console messenger:consume async -vv |