Manuel de passation des développements

De la plateforme

« SPA/RAC »

MEDIANET SARL 12, Av. Hédi Nouira, Résidence MAYA D1 ENNASR II 2037 Tunis - Tunisie
Tél.(L.G.) : +216 71 827 484 Fax : +216 71 827 464 e-mail : info@medianet.com.tn
www.medianet.com.tn

Table des matières

I. Introduction

L'objectif de ce document est de décrire l'architecture applicative et technique de la plateforme SPARAC afin de répondre aux intérêts et préoccupations de tous les principaux acteurs.

Nous allons faire recours à quelques diagrammes afin de décrire l'architecture d'un point de vue :

- Architecture applicative
- Architecture technique

Ce document décrit l'architecture prescriptive. Il identifie les composantes majeures sur lesquels se base SPARAC.

II. Terminologies

Backend : ou encore coté dorsal. Représente le code source s'exécutant sur le serveur

Frontend : ou encore coté frontal. Représente le code source s'exécutant sur le client. Le client ici désigne le navigateur utilisé par l'utilisateur pour accéder au site.

Backoffice : représente l'ensemble des interfaces visuelles (GUI) dédiées à la gestion interne de l'application

FrontOffice : représente l'ensemble des interfaces visuelles (GUI) destinées au grand public

API : sert de façade par laquelle une application offre des services à d'autres applications.

Endpoint : un point d'entrée à travers lequel on pourrait utiliser une API

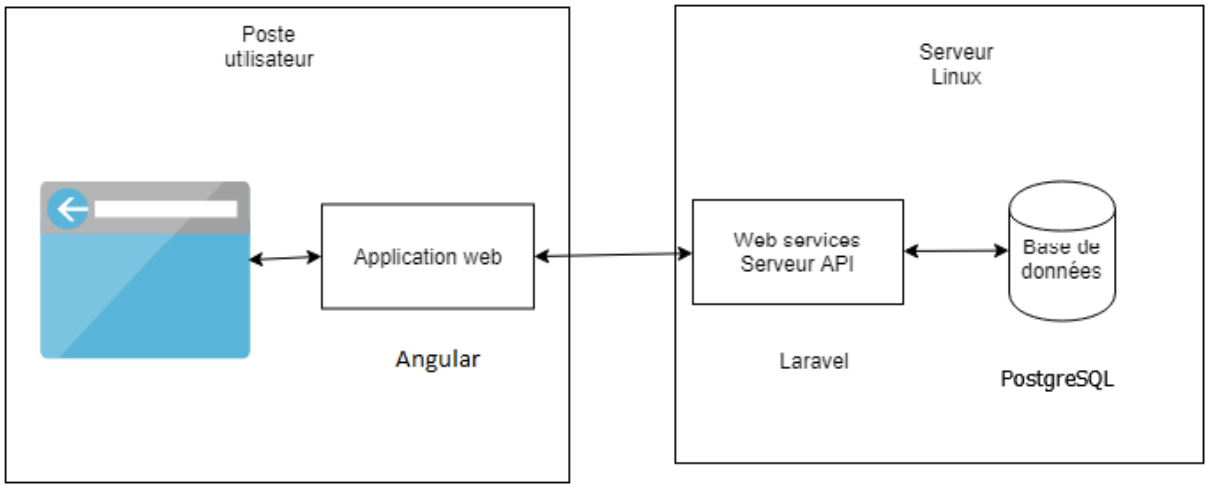
III. Architecture applicative

La plateforme SPARAC est une application n-tiers qui utilise l'approche orientée objets et elle est composée de 3 couches principales :

- La couche présentation
- La couche métier
- Et la couche de données

Le système est composé de 2 sous-systèmes. Le premier sous système représente la couche représentation et le 2^{ième} sous système contient la couche métier et base de données.

La couche représentation utilise la Framework Angular et interagie avec la couche métier à travers des requêtes http.



IV. Architecture technique

L'application se compose de 2 principaux volets. Le coté frontal est développé à l'aide de la technologie Angular et la partie dorsale est développée à l'aide de Laravel et VueJs.
La partie dorsale comporte :

- Une API webservices : Cette partie représente l'ensemble des 'endpoints' à consommer par la backoffice. une API RESTful à base Laravel.
- Le front-office : Cet espace est accessible au public et sera accessible pour tout le monde.

Ce choix d'architecture permet l'extension vers d'autre plateforme et expose les données souhaitées à des parties tiers qui auront intérêt à les incorporer dans leurs systèmes.
Le serveur API aussi désigné par la couche applicative à travers ce document, repose sur Laravel qui lui à son tour repose sur PHP 7, Apache et PostgreSQL. L'ensemble des briques composant le serveur repose sur une instance Linux. La configuration de l'application se fait à travers le fichier de configuration « .env ». l'accès public au serveur est sécurisé avec le Protocol SSL et les ressources. L'authentification implémente la norme JWT.

V. Specifications techniques

V.1 Application Web

L'application web utilise le template Angular et html Metronic qui utilise la version Angular 8.2.5. Un aperçu se trouve sur le lien suivant :
https://keenthemes.com/metronic/preview/angular/demo1/

La documentation concernant le framework Angular peut être trouvée sur le lien suivant :
https://angular.io/

V.1.1 Paquets utilisés

Les paquets suivants sont utilisés au niveau frontal :

- "@angular/animations": "^8.2.5",
- "@angular/cdk": "^8.1.1",
- "@angular/common": "^8.2.5",
- "@angular/compiler": "^8.2.5",
- "@angular/core": "^8.2.5",
- "@angular/forms": "^8.2.5",
- "@angular/http": "^7.2.15",
- "@angular/platform-browser": "^8.2.5",
- "@angular/platform-browser-dynamic": "^8.2.5",
- "@angular/platform-server": "^8.2.5",
- "@angular/router": "^8.2.5",
- "@asymmetrik/ngx-leaflet": "^6.0.1",
- "@asymmetrik/ngx-leaflet-draw": "^5.0.1",
- "@fortawesome/fontawesome-free": "^5.8.1",
- "@ng-bootstrap/ng-bootstrap": "^4.2.1",
- "@ngrx/effects": "^8.1.0",
- "@ngrx/entity": "^8.1.0",
- "@ngrx/router-store": "^8.1.0",
- "@ngrx/store": "^8.1.0",
- "@ngrx/store-devtools": "^8.1.0",
- "@ngx-loading-bar/core": "^4.2.0",
- "@ngx-translate/core": "^11.0.1",

- "@types/lodash": "^4.14.136",
- "angular-in-memory-web-api": "^0.8.0",
- "animate.css": "^3.7.2",
- "bootstrap": "^4.3.1",
- "chart.js": "^2.8.0",
- "chartist": "^0.11.3",
- "classlist.js": "^1.1.20150312",
- "core-js": "^3.1.4",
- "hammerjs": "^2.0.8",
- "highlight.js": "^9.15.8",
- "jquery": "^3.4.1",
- "leaflet": "^1.5.1",
- "leaflet-draw": "^1.0.4",
- "leaflet-draw-locales": "^1.1.1",
- "lodash": "^4.17.11",
- "material-design-icons": "^3.0.1",
- "moment": "^2.24.0",
- "ng-inline-svg": "^8.5.1",
- "ngrx-store-freeze": "^0.2.4",
- "ngx-clipboard": "^12.2.0",
- "ngx-daterangepicker-material": "^2.1.7",
- "ngx-highlightjs": "^3.0.3",
- "ngx-perfect-scrollbar": "^8.0.0",
- "ngx-permissions": "^7.0.3",
- "ngx-summernote": "^0.7.0",
- "ngx-toastr": "^10.0.4",
- "object-path": "^0.11.4",
- "owl.carousel": "^2.3.4",
- "perfect-scrollbar": "^1.4.0",
- "popper.js": "^1.15.0",
- "rxjs": "^6.5.3",
- "socicon": "^3.0.5",
- "summernote": "^0.8.12",

- "toastr": "^2.1.4",
- "tooltip.js": "^1.3.2",
- "tslib": "^1.10.0",
- "web-animations-js": "^2.3.2",
- "zone.js": "~0.9.1"

V.1.2 Règles de code Javascript

Nous utilisons les règles de codage recommandées par l'équipe Angular. Ci-dessous quelques règles que nous utilisons à travers le projet. Une liste complète des règles et des conventions se trouve sur le lien suivant : https://angular.io/guide/styleguide

- Il est préférable de redistribuer le composant et ses classes de support dans leurs propres fichiers dédiés.
- Utiliser des noms cohérents pour tous les symboles.
- Utiliser des tirets pour séparer les mots du nom descriptif pour les noms de fichier.
- Utiliser des points pour séparer le nom descriptif du type pour les noms de fichier.
- Utiliser « CamelCase » pour les noms de classe.

Nous avons aussi utilisé le REDUX paterne à travers la librairie NGRX. Pour mieux comprendre ce patterne, la lecture de la ressource suivante est recommandée : https://ngrx.io/guide/store

V.1.3 Prérequis

L'application web nécessite les prérequis suivants :

- Serveur web Apache pour gérer les requêtes
- Node et npm pour installer les dépendances si nécessaire

V.2 Serveur API

La couche applicative est représentée par le serveur API qui lui à son tour interagie avec la couche de données à travers une couche intermédiaire d'accès de données (Database Access Layer DAL)

V.2.1 Framework utilisé

Le serveur des services web utilise le framework Laravel qui se base sur PHP. La version Laravel utilisée est la version 6.2. (https://laravel.com/)

V.2.2 Prérequis serveur web

L'application tourne sous un serveur web apache et utilise les modules de réécriture de chemin d'accès et l'accès est sécurisé par le protocol SSL.

L'installation des dépendances se fait à travers composer et la gestion du serveur à distance nécessite l'activation du port ssh.

V.2.3 Dépendances

Les dépendances au niveau serveur Apache sont comme suite :

- php7_module
- rewrite_module
- ssl_module

Les modules nécessaires au bon fonctionnement de Laravel doivent être installés et activés.

Ci-dessous les extentions php à installer et activer :

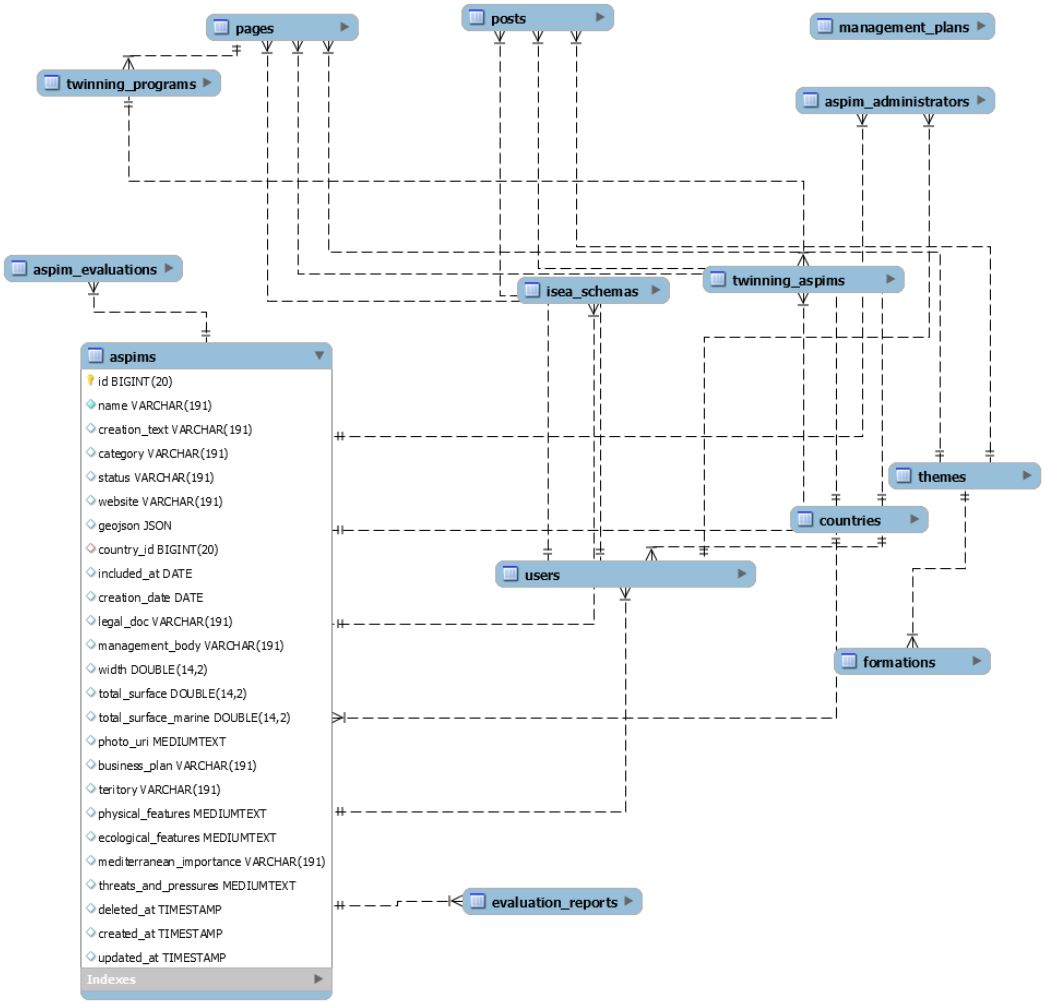
- PHP >= 7.1.3
- BCMath PHP Extension
- Ctype PHP Extension
- JSON PHP Extension
- Mbstring PHP Extension
- OpenSSL PHP Extension
- PDO PHP Extension
- Tokenizer PHP Extension
- XML PHP Extension

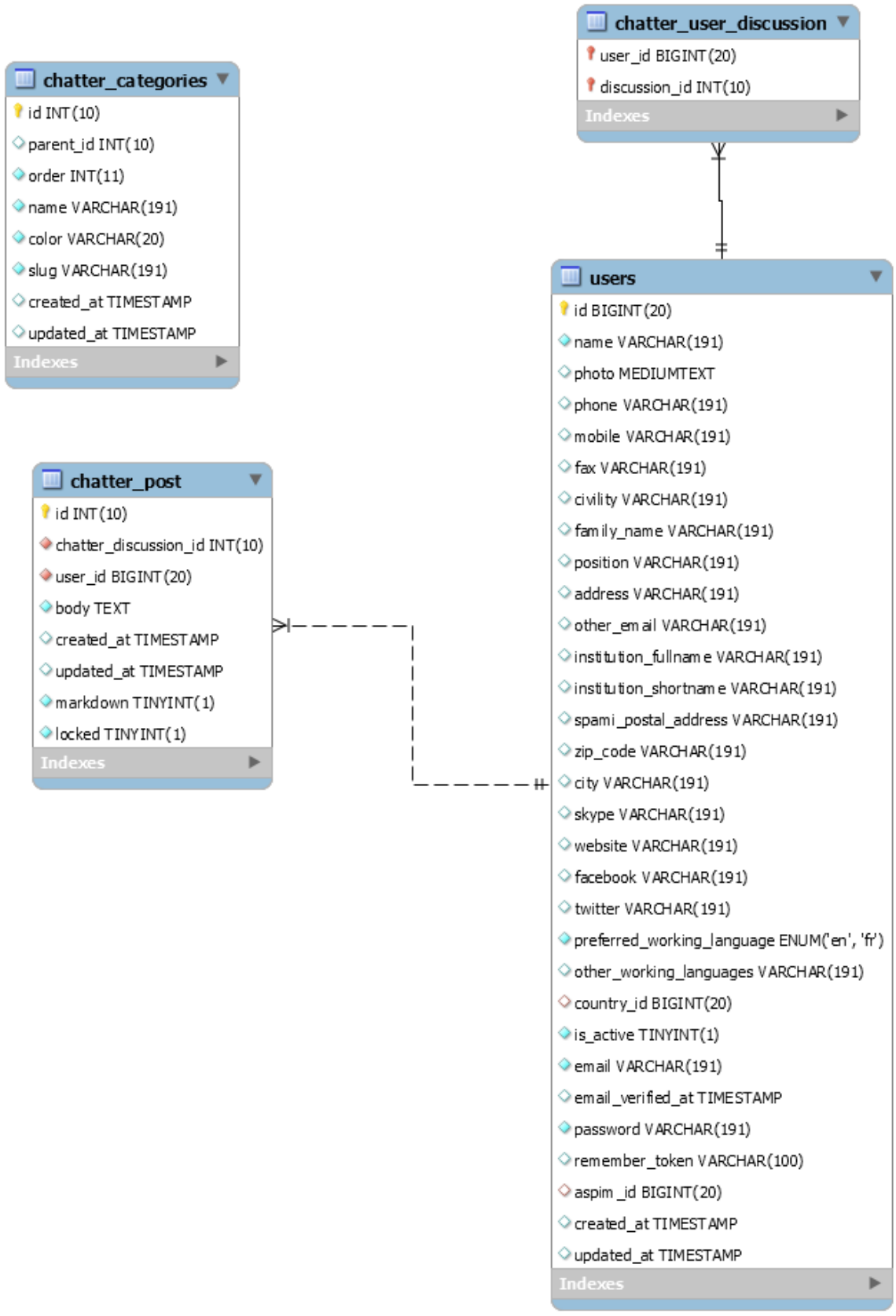
V.2.4 Couche de données

Le serveur API utilise une base de données PostgreSQL. La version (9.4-9.6), 10, et 11 peuvent être employées, avec l'extension POSTGIS 2.5 installée et activée.

VI. Modèle physique des données (MPD)

Ci-dessous un aperçu sur le modèle physique des données.




chatter_user_discussion
user_id BIGINT(20)
discussion_id INT(10)
Indexes
chatter_categories
id INT(10)
parent_id INT(10)
order INT(11)
name VARCHAR(191)
color VARCHAR(20)
slug VARCHAR(191)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
users
id BIGINT(20)
name VARCHAR(191)
photo MEDIUMTEXT
phone VARCHAR(191)
mobile VARCHAR(191)
fax VARCHAR(191)
civility VARCHAR(191)
family_name VARCHAR(191)
position VARCHAR(191)
address VARCHAR(191)
other_email VARCHAR(191)
institution_fullname VARCHAR(191)
institution_shortname VARCHAR(191)
spami_postal_address VARCHAR(191)
zip_code VARCHAR(191)
city VARCHAR(191)
skype VARCHAR(191)
website VARCHAR(191)
facebook VARCHAR(191)
twitter VARCHAR(191)
preferred_working_language ENUM('en', 'fr')
other_working_languages VARCHAR(191)
country_id BIGINT(20)
is_active TINYINT(1)
email VARCHAR(191)
email_verified_at TIMESTAMP
password VARCHAR(191)
remember_token VARCHAR(100)
aspim_id BIGINT(20)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
chatter_post
id INT(10)
chatter_discussion_id INT(10)
user_id BIGINT(20)
body TEXT
created_at TIMESTAMP
updated_at TIMESTAMP
markdown TINYINT(1)
locked TINYINT(1)
Indexes

VII. Modèle conceptuel des données (MCD)

Le modèle conceptuel des données est représenté ci-dessous