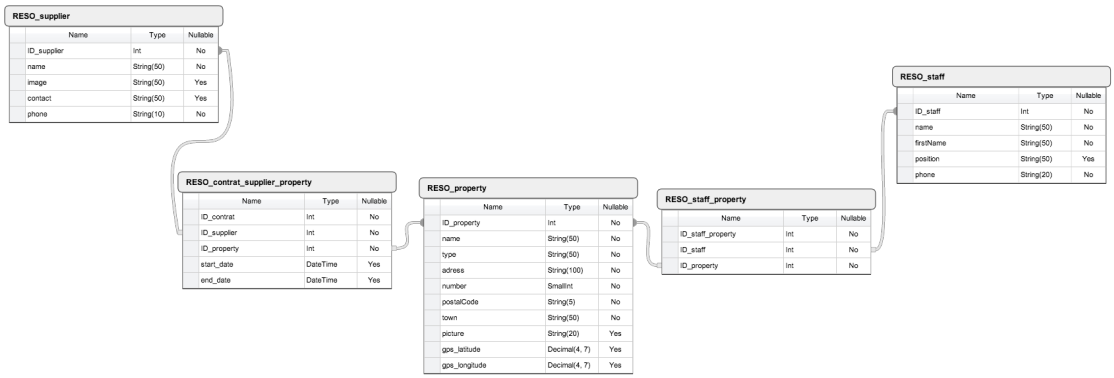
Code RESO

1 Table des matières

2 Base de données

La base données a été développées sous MySql. Rappel :

- Le varchar doit obligatoirement comporter un nombre par défaut (maximum de caractère).
- Collation : ce sont les règles de comparaison des caractères.



2.1 Paramètres de connection :

Les paramètres ci-dessous ne sont valable qu'à partir d'un server OVH.

- Serveur : mysql5-8.perso
- Utilisateur : autogel
- Nom de la base : autogel
- Mot de passe : 4W3r3a8L

2.2 Table : RESO_property

La table RESO_property contient les caractéristique générale d'une propriété.

Nom	Caractéristiques
ID_property	Clé primaire de la table
name	nom de la propriété
code	nomenclature
type	Varchar pour type d'immeuble : villa, appartement....
adress	Adresse sans le numéro
number	Numéro de l'adresse
postalCode	Code postale
town	commune
picture	nom du fichier photo. Attention : le chemin est défini dans le php
gps_latitude	coordonnées gps pour google maps
gps_longitude	coordonnées gps pour google maps

2.3 Table : RESO_room

Cette table contient les locaux des propriétés.

champ	caractéristique
ID_room	clé primaire
name	nom du local
code	servira pour la nomenclature
floor	etage
department	département
ID_property	bâtiment auquel il appartient

2.4 Table : RESO_staff

La table RESO_staff reprend le personnel chargé de la gestion des propriétés.

champ	caractéristique
ID_staff	clé primaire
name	nom de famille
firstName	prénom
position	poste
phone	téléphone fixe

2.5 Table : RESO_staff_property

La table RESO_staff_property fait la relation « many to many » entre la table RESO_staff et RESO_property. Le moteur de stockage est innoDB pour la gestion des clés étrangères.

champ	caractéristique
ID_staff_property	clé primaire
ID_staff	clé étrangère
ID_property	clé étrangère

2.6 Table : RESO_supplier

La table RESO_supplier reprend tous les fournisseurs.

champ	caractéristique
ID_supplier	clé primaire
name	nom de la société
image	emplacement du fichier contenant le logo
contact	nom de la personne de contact au sein de la société
phone	téléphone de la personne de contact au sein de la société

2.7 Table : RESO_contract_supplier_property

La table RESO_contract_supplier_property fait le lien « many to many » entre les fournisseur et les propriétés.

champ	caractéristique
ID_contract	clé primaire
ID_supplier	clé étrangère des fournisseur
ID_property	clé étrangère des propriétés
start_date	date de commencement du contrat
end_date	date de fin de contrat

2.8 Table : RESO_fire_asset_type

La table RESO_fire device_type contient le type d'élément succeptible d'être utilisé dans la prévention incendie. Cette table permet de remplir la table RESO_fire_device.

champ	caractéristique
ID_fire_asset_type	clé primaire
name	nom de l'élément utilisé dans la prévention incendi
description	description du device
ID_preventive_maintenance_plan	check if valuable

2.9 Table : RESO_fire_asset

La table RESO_fire_device contient tous les éléments de prévention incendie contenus dans notre patrimoine. Cette table permet de remplir la table RESO_fire_device.

champ	caractéristique
ID_fire_asset	clé primaire
code	identifiant suivant la nomenclature définie (permettant de savoir où se situe le device.
ID_fire_asset_type	type du device, issue de la table RESO_fire_device_type (foreign key)
ID_room	local où se trouve l'asset
name	name
description	description du device
install_date	date de l'installation
purchase_date	date de l'achat

length_life_type	C'est un enum : [day, month, year]
length_life	durée de vie
ID_asset_status	status foreign key
image	nom fichier dans le répertoire /images/fire/
model	
ID_manufacturer	fabriquant (fk)
ID_supplier	fournisseur (foreign key)
serial	
ID_preventive_maintenance_plan	check if valuable

2.10 Table : RESO_asset_status

Cette table permet de définir le statut d'un asset : actif

champ	caractéristique
ID_asset_status	clé primaire
nom	

2.11 Table : RESO_manufacturer TODO

Cette table contient les fabriquants de nos actifs.

champ	caractéristique
ID_manufacturer	clé primaire
nom	

2.12 Table : RESO_maintenance_type TODO

La table contiendra :

- corrective
- inspection
- maintenance

champ	caractéristique
ID_maintenance_type	clé primaire
nom	

2.13 Table : RESO_maintenance_priority TODO

La table contiendra :

- highest : today
- high : tomorow
- medium : within 2 days
- low : within 1 week
- lowest : in 1 month

champ	caractéristique
ID_maintenance_type	clé primaire

nom

2.14 Table : RESO_work_order TODO

La table contiendra les demandes de travail via tickets. Des works order seront générés pour chaque asset sur base des paramètres suivants :

- preventative maintenance plan
- length life of the asset

champ	caractéristique
ID_work_order	clé primaire
nom	
ID_asset	c'est pour le asset concerné
ID_maintenance_type	
ID_maintenance_priority	
description	
ID_work order status	
schedule_date	date prévue
ID_accountable	

2.15 Table : RESO_work_order_status TODO

La table contiendra le statut des ordres de travail :

- open
- closed – completed
- closed – uncompleted
- holding
- in progress
- assigned
- requested : demande d'ouverture de ticket.

champ	caractéristique
ID_work_order_status	clé primaire
nom	

2.16 Table : RESO_preventative_maintenance_plan TODO

Cette table contient tous les plan d'entretien préventif, par exemple un constructeur recommande d'entretenir un groupe électrogène chaque 6 mois.

champ	caractéristique
ID_preventive_maintenance	clé primaire
nom	
frequency_type	jours, mois, an, utilisation
frequency	fréquence maintenance (en frequency_type)
activation_date	date d'introduction du plan de maintenance
end_date	date à laquelle ce plan n'est plus actif (status not in use)
description	description de la maintenance
domain	feu, electricité, hvac, sanitaire

status	
instruction	ce qu'il faut faire pour la maintenance

2.17 Table : RESO_tasks TODO

Cette table contient les tâches à effectuer lors d'un entretien de maintenance.

champ	caractéristique
ID_task	clé primaire
nom	
description	
ID_preventative_maintenance	
estimated time	

3 Code

Le programme est en php avec le framework codeigniter. La plateforme de test est à l'adresse suivante :

- http://www.autogel.be/RESO/

3.1 Syntaxe

J'ai adopté les règles de syntaxe suivantes :

- f_xxxxx : pour les méthodes des classes.
- c_xxxxx : pour les controllers.
- v_xxxxx : pour les views.
- m_xxxxx :pour les models (db).

3.1.1 Rappel

La triple égalité === permet de vérifier que les opérants sont égaux et de même type.

3.2 configuration

La configuration se fait par le biais de fichiers se trouvant dans le répertoire / application/config.

3.2.1 calendar

Un fichier de configuration nommé calendar.php est utilisé pour l'utilisation du calendrier. Deux paramètres sont définis :

- day_type : long : permet d'afficher les jours en entier.
- template : permet de définir les div et les span pour le rendu qui sera explicité dans le fichier calendar.css

3.2.2 database

La configuration de la database avec les paramètres de connections se fait dans le fichier suivant : /application/config/database.php

3.2.3 Routes

Le fichier /application/config/routes.php permet de définir le controller par défaut et le et les méthodes par défaut. Pour rappel les URL dans codeigniter fonctionnent sur base de trois segments:

exemple.com/classe_controller/méthode_controller/paramètre_méthode

Les routes dans codeigniter fonctionne de la façon suivante :

$route['property/(:any)'] = 'reso/property/$1';

lorsqu'une URI correspond exactement à une key (dans l'exemple ci-dessus « property/ (:any) ») de notre array $route, codeingniteur transforme cette URI en la valeur de la key (dans notre exemple : « reso/property/$1 »).

(:any) correspond à une chaine de caratère.
(:num) correspond à un nombre.

3.2.3.1 c_insert/f_insert_staff

```
$route['c_insert/f_insert_staff'] = 'c_insert/f_insert_staff';
```

3.2.3.2 c_insert/f_insert

```
$route['c_insert/f_insert'] = 'c_insert/f_insert';
```

3.2.4 encryption key

Pour pouvoir utiliser les variables de session (pour s'assurer que l'utilisateur est logged et qu'un bâtiment est sélectionné), il faut définir une encryption key (clé de cryptage) dans le fichier :

/application/config/config.php

La clé définit est la suivante : « rachidjoelaziz ».

3.3 Controllers

Les controllers contiennent toute la logique de fonctionnement.

3.3.1 c_reso.php

Le controller est le point central de notre application. Le controller est définit dans un fichier qui est situé dans le répertoire :

application/controllers/reso.php

On créée notre propre classe qui hérite de la classe controller. L'appel de notre controller se fait via l'url :

http://autogel.be/RESO/reso/home/[arguments]

La méthode notre url correspond à la méthode de la classe ci-dessous. Et l'argument correspond au paramètre de la méthode ci-dessous.

3.3.1.1 Variables de session

Les variables de session utilisées sont :

- ID_property : clé primaire d'une propriété, qui sera utilisée par tous les sous menu.
- logged_in : booléen pour vérifier si un utilisateur est loggé.

Les variables de session sont activées dans c_reso/f_property. Les variables de session sont désactivées dans c_reso/f_home.

3.3.1.2 __construct

Le constructeur __construct charge la librairie :

- calendar : qui permet d'utiliser les calendrier de code igniter avec les fichiers suivant :
    - config/calendar.php : fichier qui donne les paramètre de template et nom de jour.
    - css/calendar.css : fichier qui définit le css.

3.3.2 c_contrats.php

Le controller c_contrats.php gère les pages des contrats. Il utilise les vues suivantes :

- v_contrats.php

Il utilise les méthodes suivantes :

- f_contrats : elle appelle la vue v_contrats qui contient les trois types de contrat : HVAC, sécurité et maintenance.

On récupère dans la base de donnée le bâtiment concerné à l'aide de la variable de **session**.

3.3.3 c_insert.php

Le controller c_insert.php est dédié aux insertions de datas dans la db. Les fonctions du controller sont :

- f_insert_staff : cette fonction est dédiée l'insertion de nouveaux employés. Nous associons les employés à des bâtiments, par conséquent on utilise le model m_staff.php. La fonction utilise la vue : v_insert_staff.php
- f_insert_room : cette fonction est dédiée à l'insertion des nouveaux locaux. Nous utilisons la classe m_room.php pour insérer dans la RESO_room. La fonction utilise la vue : v_insert_room.php.
- f_insert() : permet de sélectionner parmi les fonctions ci-dessus via la vue : v_insert.php.

3.3.4 c_inventaire.php

La classe c_inventaire.php permet, dans sa version actuelle, d'afficher tout le contenu de la db via la fonction f_inventaire.
La fonction f_invetaire appelle la vue v_inventaire.php en lui passant la liste des propriétés et la liste des employés.

3.4 Views

Trois vues ont été créés dans le répertoire /application/views/ :

- header.php : correspond à l'entête d'une page html.
- view.php : correspond à la balise body d'une page html.
- property.php :

- footer.php : correspond à la balise footer d'une page html.

3.4.1 v_header.php
La vue v_header est appelé par :
- c_reso / f_home

Elle ne contient qu'un bouton de menu.

3.4.2 v_header_property.php
La vue v_header_property est appelée par :
- c_reso / f_property
- c_contrats / f_contrats

Elle est dédiée à ces deux fonctions car :
- googlemaps ne doit être appelée que par f_property
- les sous menus ne doivent apparaître qu'après sélection de la propriété.

Le code php avec $this->uri->segment(n) permet de vérifier quelle vue a été appelée par le controleur pour lancer le script de google maps.

Cette vue charge le css du calendrier : calendar.css

3.4.2.1 div header box
Les header_box contiennet les images des boutons du menu supérieur gauche. Ces images faisant 80px x 80px, il faut un pading du haut vers le bas de 8px car le wrapper dans lequel ils sont contenus fait 88px.

3.4.3 v_property.php
C'est la vue qui génère la fiche de synthèse de la propriété.
Elle génère aussi le calendrier.

3.4.4 v_insert_staff.php
C'est la vue qui est appelée par la fonction c_insert.php/f_insert_staff.

Pour récupérer la liste des batiments sur lesquels un employé est responsable on utilise les checkbox avec un array : property_selected.

3.4.5 v_insert_room.php
C'est la vue qui est appelée par la fonction c_insert.php/f_insert_room.

3.4.6 v_inventaire.php
La vue v_inventaire.php est appellée par c_inventaire/f_inventaire. Elle permet d'afficher le contenu des tables employés et propriétés via le helper table.

3.5 Model
Le répertoire /application/model/ contient toutes les classes nous permettant d'interagir avec la base de données :
- m_staff.php
- m_property.php

3.5.1 m_property.php
La classe m_property.php permet de gérer l'interraction avec les tables qui traitent de la property.
Les fonctions sont les suivantes :

- m_f_get_property(ID_property) : nous renvoie toutes les propriétés si aucune clé n'est donnée en paramètre. Dans le cas où un paramètre est passé, la fonction renvoie la propriété avec l'ID.

3.6 m_staff.php

La classe m_staff.php contient toutes les fonctions relatives au staff :

- f_get_staff : fonction qui retourne les employés responsable d'une propriété (qui est passée en paramètre).
- m_f_insert_staff : fonction qui permet d'insérer un nouvel employé sur base de ce qui est envoyé en $post(via formulaire). L'insertion se fait en deux phases, sachant qu'il y a deux tables : RESO_staff et RESO_staff_property (qui fait le lien entre l'employé et les bâtiments dont il est responsable). Il faut donc d'abord insérer le nouvel employé pour récupérer son numéro d'ID_staff pour insérer les nouveaux bâtiments liés. Cette fonction est appelé par c_insert/f_insert_staff.

3.7 m_room.php

Le classe m_room.php contient toutes les fonctions nécessaire à la table RESO_room :

- m_f_get_room : permet de récupérer les rooms, si on passe le paramètre ID_property : récupère les rooms d'une property.
- m_f_insert_room : permet d'insérer les nouvelles rooms via le formulaire : c_insert/f_insert_room.
- m_f_room_fields() : permet des récupérer les noms des champs de la table.

3.8 m_fire_asset

La m_fire_asset.php contient les fonctions permettant d'interagir avec la table RESO_fire_asset. Le traitement du format date est important pour l'insertion dans mysql.

4 TODO

- sécuriser le post du premier formulaire (loggin).
- encadrer les images de la page active dans le menu supérieur gauche
- faire une ligne du temps ou calendrier avec information
- interdire l'accès au page sans loggin.
- faire la validation formulaire pour insert (staff, contract).
- htaccess : configurer le meilleur htaccess (compatible) avec routes.
- intégrer les conditions sur les champs (format email, téléphone,....) d'un form dans un fichier de config.
- sécuriser tous les formulaires (logged + hidden input to interact with the model).
- upload d'images dans les formulaires d'insertion

4.1 Ticketing (demande d'intervention = work request)

On doit pouvoir:

- sélectionner le lieu (bâtiment, etage, bureaux...)
- sélectionner l'actif concerné
- sélectionner le type de problème
- sélectionner le demandeur
- sélectionner la priorité
- générer des statistiques sur les nombres de tickets
- consulter le statut

- générer un rapport d'intervention avec : durée – intervenants - ...
-

5 Fonctionnalités GMAO

- la gestion de stock : prend en compte les pièces détachés.
- gestion des actifs : sites – zone – etage – local – actif – composants (voir stock ci-dessus).
- process : actif – plan de maintenance – bon de travail
- les demandes d'intervention (work request) diffère d'un ordre de travail

6 Lessons from the project

6.1 MySql

Pour intégrer des clés etrangères il faut :

- innoDB
- au moins un index dans la table d'origine de la clé étrangère.

7 Index