Le module évènement - Documentation technique

1.La structure de données

Structure de la table agenda

```sql
CREATE TABLE agenda
(
    id       int auto_increment primary key,
    titre    varchar(150) not null,
    date     date not null,
    contenu  text not null

);
```

Contenu

id	titre	date	contenu
1	run2k challenge	2025-09-11	Pour inaugurer la nouvelle saison de piste, venez ...
2	Demi-finale des championnats de France de 5km	2025-09-22	Organisés par le VDS au parc de la Hotoie (parcour...
3	100km de la Somme, Marathon d' Amiens Metropole et...	2025-10-12	Organisée par le VDS avec départ et arrivée au Gra...
4	Finale des 4 saisons	2025-11-03	Traditionnelle édition finale d'automne, avec son ...

Déclencheur

```sql
CREATE TRIGGER before_insert_agenda
    BEFORE INSERT ON agenda
    FOR EACH ROW
BEGIN
    -- Si l'ID est NULL ou déjà existant ou mal renseigné (<= 0)
    IF NEW.id not regexp '^[0-9]+' OR EXISTS (SELECT 1 FROM agenda WHERE id = NEW.id) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'L''identifiant n''est pas valide';
    END IF;

-- Trigger sur le contenu
    IF NEW.contenu IS NULL OR LENGTH(NEW.contenu) = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Le contenu ne peut pas être vide.';
    END IF;

    SET NEW.contenu = REPLACE(NEW.contenu, '<', '&lt;');
    SET NEW.contenu = REPLACE(NEW.contenu, '>', '&gt;');

-- Trigger sur le titre
    IF NEW.titre IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Le titre ne peut pas être nul';
    ELSEIF LENGTH(NEW.titre) < 3 OR LENGTH(NEW.titre) > 100 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Le titre doit contenir entre 3 et 100 caractères';
    ELSEIF NEW.titre NOT REGEXP '^[A-Za-z0-9,'' -]+$' THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Le titre doit contenir uniquement des lettres, des chiffres, des virgules, des espaces et des tirets';
    END IF;

-- Trigger sur la date
    IF NEW.date IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'La date ne peut pas être nulle';
    ELSEIF NEW.date NOT REGEXP '^[0-9]{4}-[0-9]{2}-[0-9]{2}$' THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'La date doit être au format YYYY-MM-DD';
    ELSEIF NEW.date <= CURDATE() THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'La date doit être supérieure à la date du jour';
    END IF;
END;
```

```sql
CREATE TRIGGER before_update_agenda
    BEFORE UPDATE ON agenda
    FOR EACH ROW
begin
    -- Vérifie si l'ancien ID est différent du nouveau et empêche la modification
    if old.id != new.id then
        signal sqlstate '45000' set message_text = 'Modification du champ ID interdite';
end if;

        -- Trigger sur le contenu
        if new.contenu is null or LENGTH(new.contenu) = 0 then
            signal sqlstate '45000' set message_text = 'Le contenu ne peut pas être vide.';
        end if;

        set new.contenu = REPLACE(new.contenu, '<', '&lt;');
        set new.contenu = REPLACE(new.contenu, '>', '&gt;');

-- Trigger sur le titre
        if new.titre is null then
            signal sqlstate '45000' set message_text = 'Le titre ne peut pas être nul';
        elseif LENGTH(NEW.titre) < 3 OR LENGTH(NEW.titre) > 100 then
            signal sqlstate '45000' set message_text = 'Le titre doit contenir entre 3 et 100 caractères';
        elseif NEW.titre not regexp '^[A-Za-z0-9,'' -]+$' then
            signal sqlstate '45000' set message_text = 'Le titre doit contenir uniquement des lettres, des chiffres, des virgules, des espaces et des tirets';
        end if;

-- Trigger sur la date
        if new.date is null then
            signal sqlstate '45000' set message_text = 'La date ne peut pas être nulle';
        elseif new.date not regexp '^[0-9]{4}-[0-9]{2}-[0-9]{2}$' then
            signal sqlstate '45000' set message_text = 'La date doit être au format YYYY-MM-DD';
        elseif new.date <= CURDATE() then
            signal sqlstate '45000' set message_text = 'La date doit être supérieure à la date du jour';
    end if;
end;
```

1. **La classe métier agenda**

```php
class Agenda extends Table
{
    no usages
    public function __construct()
    {
        parent::__construct( nomTable: 'agenda');

        // seuls les colonnes pouvant être modifiées par l'administrateur sont définies

        // Nom (titre)
        $input = new inputText();
        $input->Require = true;
        $input->SupprimerAccent = false;
        $input->SupprimerEspaceSuperflu = true;
        $input->Pattern = "^[a-zA-Z0-9À-ÖØ-öø-ÿ' -]{10,150}$";
        $input->MaxLength = 150;
        $this->columns['titre'] = $input;

        // Date
        $input = new inputDate();
        $input->Require = true;

        // la date ne doit pas être inférieure à la date du jour ni dépassé d'un an la date du jour
        $input = new InputDate();
        $input->Min = date( format: 'Y-m-d');
        $input->Max = date( format: "Y-m-d", strtotime( datetime: "+1 year"));
        $this->columns['date'] = $input;

        // Contenu
        $input = new InputText();
        $input->MinLength = 10;
        $input->MaxLength = 1000;
        $input->Require = true;
        $this->columns['contenu'] = $input;
    }
```

```php
    // ------------------------------------------------------------------------
    // Méthodes statiques relatives aux opérations de consultation
    // ------------------------------------------------------------------------

    public static function getAll(): array
    {
        // Récupération des paramètres du téléversement
        $sql = <<<EOD
        Select id, titre, date, contenu, DATE_FORMAT(date, '%d/%m/%Y') as dateFr
        From agenda
        order by date
EOD;
        $select = new Select();
        return $select->getRows($sql);
    }



    /**
     * Récupère les données d'un évènement à partir de son id
     * @param int $id
     * @return array|false
     */
    public static function getById($id)
    {
        $sql = <<<EOD
             Select id, titre, contenu, date
             From agenda
             where id = :id;
EOD;
        $select = new Select();
        return $select->getRow($sql, ['id' => $id]);
    }
}
```

2. L'affichage des évènements sur la page d'accueil

2.1 Le fichier index.php

```php
// Récupération des événements de l'agenda : titre, date, contenu
$select = new Select();
$sql = <<<EOD
    SELECT titre, date, contenu
    FROM agenda
EOD;

$data = json_encode($select->getRows($sql));

$head = <<<EOD
<script>
    let data = $data;
</script>
EOD;
```

3.2. Le fichier index.js

```
// Affichage des événements de l'agenda dans des cartes
/* global data */

// création d'une div 'row' de class row
const row : HTMLDivElement  = document.createElement( tagName: 'div');
row.classList.add('row');

// parcourir le tableau d'objet data
for (const element of data) {
    // définir une div 'col' utilisant les classes 'col-' afin d'avoir 4 cadres sur un grand écran, 3 sur un lg ou md, 2 sur un sm et 1 en dessous
    const col : HTMLDivElement  = document.createElement( tagName: 'div');
    col.classList.add('col-xl-3', 'col-lg-4', 'col-sm-6');

    // définir une div 'carte'
    const carte : HTMLDivElement  = document.createElement( tagName: 'div');
    carte.classList.add('card', 'mb-4');

    // définir l'entête qui contient le titre et la date de l'événement
    const entete : HTMLDivElement  = document.createElement( tagName: 'div');
    entete.classList.add('card-header', 'bg-dark', 'text-white');

    const titre : HTMLHeadingElement  = document.createElement( tagName: 'h5');
    titre.classList.add('card-title');
    titre.innerText = element.titre;

    const date : HTMLHeadingElement  = document.createElement( tagName: 'h6');
    date.classList.add('card-subtitle', 'mb-2', 'text-white');
    date.innerText = element.date;

    entete.appendChild(titre);
    entete.appendChild(date);

    // définir le corps qui contient le contenu de l'événement
    const corps : HTMLDivElement  = document.createElement( tagName: 'div');
    corps.classList.add('card-body');
    corps.innerHTML = element.contenu;
```

```
    // assembler les deux éléments composant la carte : entete - corps
    carte.appendChild(entete);
    carte.appendChild(corps);

    // placer la carte dans la colonne
    col.appendChild(carte);

    // placer la colonne dans la ligne
    row.appendChild(col);
}

// intégrer l'ensemble sur l'interface
document.getElementById( elementId: 'lesCartes').appendChild(row);
```

3. L'administration des évènements

4.1. Le fichier administration/evenement/index.php

```php
// Contrôle de l'accès
Administrateur::controlerAcces();

// génération d'un token pour garantir l'origine des appels vers les autres scripts du module
$token = Jeton::creer();

// chargement des données
$data = json_encode(Agenda::getAll());

$head =<<<EOD
<script >
    let data = $data
     const token = '$token';
</script>
EOD;
```

4.2. Le fichier administration/evenement/index.js

```javascript
const lesLignes : HTMLElement  = document.getElementById( elementId: 'lesLignes');
const msg : HTMLElement  = document.getElementById( elementId: 'msg');

lesLignes.innerHTML = '';
const row : HTMLDivElement  = document.createElement( tagName: 'div');
row.classList.add('row');
for (const element of data) {

    let id = element.id;
    let tr : HTMLTableRowElement  = lesLignes.insertRow();
    tr.style.verticalAlign = 'middle';
    tr.id = id;

    // colonne contenant des boutons d'action
    let td : HTMLTableCellElement  = tr.insertCell();

    let i : HTMLElement  = document.createElement( tagName: 'i');
    i.classList.add('bi', 'bi-x', 'm-1');
    i.style.color = 'red';
    i.title = "Supprimer l'évènement";
    i.onclick = () => confirmer(() : void  => supprimer(element.id));
    td.appendChild(i);
    i = document.createElement( tagName: 'i');
    i.classList.add('bi', 'bi-pencil-square', 'text-warning', 'm-1');
    i.title = "Modifier l'évènement";
    i.onclick = () : void  => {
        location.href = "modification.php?id=" + element.id;
    };
    td.appendChild(i);

    // affichage de la date
    td = tr.insertCell();
    td.style.textAlign = "center";
    td.innerText = element.dateFr;

    // affichage du titre
    tr.insertCell().innerText = element.titre;
```

```
1 usage   new *
function supprimer(id : int ) : void  {
    $.ajax({
        url: '/ajax/supprimer.php',
        method: 'POST',
        data: {
            table : 'agenda',
            id: id,
            token : token
        },
        dataType: 'json',
        success: data => {
            if (data.success) {
                afficherSucces(data.success);
                // Mise à jour de l'interface
                const ligne : HTMLElement  = document.getElementById(id);
                ligne.parentNode.removeChild(ligne);
            } else {
                for (const key in data.error) {
                    const message = data.error[key];
                    if (key === 'global') {
                        msg.innerHTML =  genererMessage(message);
                    } else  {
                        afficherErreur('Une erreur est survenue lors de la suppression');
                        console.error(message);
                    }
                }
            }
        },
        error: reponse => {
            afficherErreur('Une erreur imprévue est survenue');
            console.log(reponse.responseText);
        }
    });
}
```

4.3. Le fichier administration/evenement/ajout.php

```
// Contrôle de l'accès
Administrateur::controlerAcces();

// génération d'un token pour garantir l'origine des appels vers les autres scripts du module
$token = Jeton::creer();

// chargement des données utilisées par l'interface
$min = date( format: 'Y-m-d');

$head =<<<EOD
<script>
    const token = '$token';
    const min = '$min';
</script>
EOD;
```

4.4. Le fichier administration/evenement/ajout.js

```
function ajouter() : void  {
    $.ajax({
        url: '/ajax/ajouter.php',
        type: 'POST',
        data: {
            table : 'agenda',
            titre: titre.value,
            date: date.value,
            contenu: contenu.value,
            token: token
        },
        dataType: 'json',
        success: (data) : void  => {
            if (data.success) {
                retournerVers("L'évènement a été ajouté", '.');
            } else {
                for (const key in data.error) {
                    const message = data.error[key];
                    if (key === 'system') {
                        afficherErreur('une erreur inattendue est survenue');
                    } else if (key === 'global') {
                        msg.innerText = message;
                    } else {
                        afficherErreurSaisie(key, message);
                    }
                }
            }
        },
        error: reponse => {
            afficherErreur('Une erreur imprévue est survenue');
            console.log(reponse.responseText);
        }
    });
}
```