

**Revivre l'événement :**

# Sommaire

<b>Présentation générale de l'application :</b>	<b>2</b>
Objectif de l'application :	2
Interactivité de l'application :	2
<b>Modèle de données :</b>	<b>3</b>
<b>Structure de l'application :</b>	<b>4</b>
<b>Points d'améliorations</b>	<b>13</b>

# I.Présentation générale de l'application :

## A. Objectif de l'application :

La finalité de cette application web est de permettre à toute personne qui le souhaite de “revivre” un événement qui a eu lieu. Pour ce faire, l'utilisateur disposera d'une carte qui localise les événements disponibles sur l'application et une frise chronologique qui répertorie l'ensemble des sous-événements qui sont inclus dans l'événement principal.

Une fois qu'un événement a été sélectionné l'application permet à l'internaute d'avoir accès à un grand nombre d'éléments, allant de la description de l'événement, jusqu'à l'ensemble des items collectés (tweets relatant les faits, photos/vidéo/audio illustrant les faits), en passant par sa durée et sa position affichée sur Google Maps.

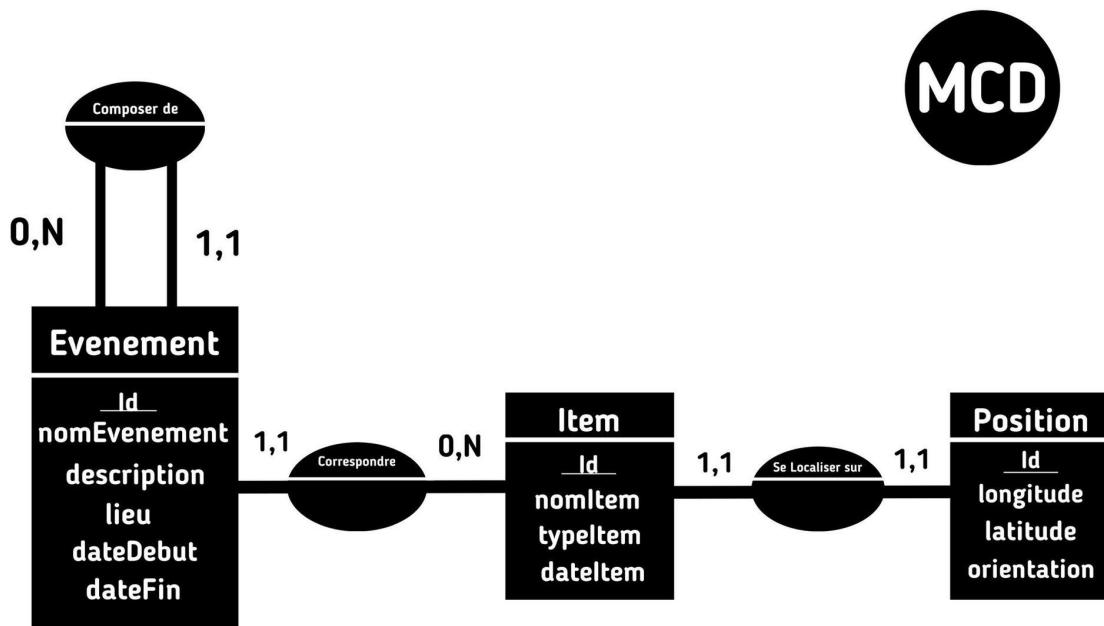
## B. Interactivité de l'application :

Les internautes ne peuvent pas uniquement interagir avec l'application afin de revivre un événement, ils peuvent aussi créer un nouvel événement.

En effet, l'application comprend un wiki permettant à n'importe qui d'ajouter un événement en renseignant le nom de l'événement, une description de ce dernier, le lieu où il s'est déroulé et ses dates de début et de fin. Une fois créé, un événement peut être complété par divers items que d'autres utilisateurs ayant participé au même événement possèdent.

## II. Modèle de données :

Modèle Conceptuel de Données :

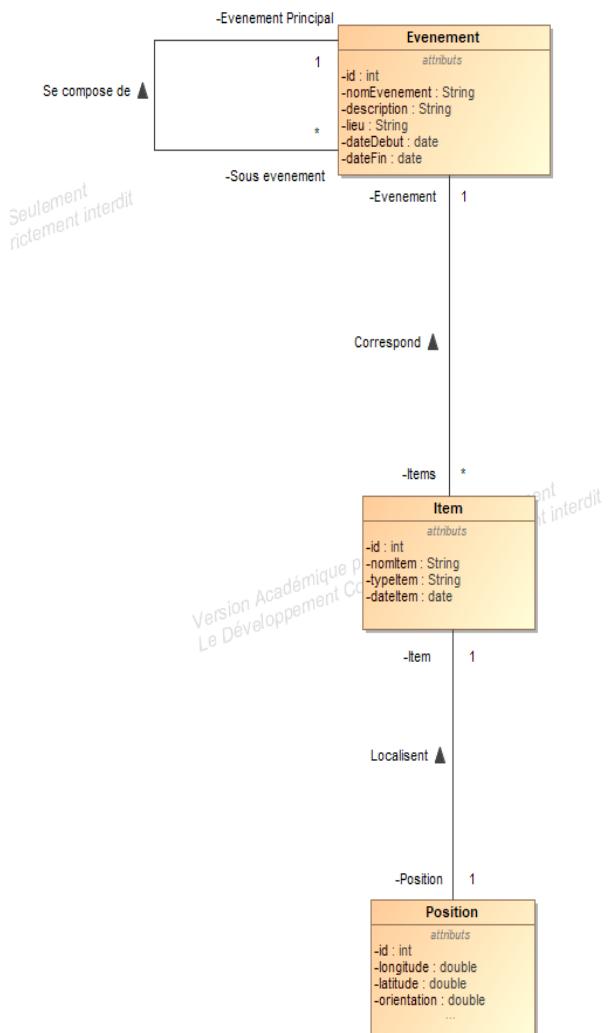


Pour notre application web on utilise des événements. Ces événements peuvent être composés d'autres événements, on aura alors un événement principal constitué de sous événements.

Une fois l'événement créé on peut ajouter des items (photo, vidéo...) qui correspondent à l'événement en question.

Ces Items, en plus de leur nom, leur type, leur date, ont une position qui donne l'emplacement géographique où a été produit l'item (à l'aide de sa longitude et de sa latitude) ainsi que l'orientation avec laquelle il a été produit (c'est à l'angle formé entre la position de l'événement et celle de l'item).

## Diagramme de classe :



Dans ce diagramme de classe, la classe Evenement correspond à l'événement dans son ensemble (exemple la coupe du monde 2018 est un Evenement Principal qui a eu lieu en Russie entre le 14 juin et le 15 juillet 2018 et la Finale est un Sous Evenement).

Plusieurs Items peuvent correspondre à un Evenement

Un Item peut correspondre à un tweet, une photo, une vidéo, un audio relatant un fait de l'Evenement en question selon un certain point de vue à un certain moment.

La classe Position permet d'avoir la localisation d'un item en indiquant sa longitude, sa latitude et une possible orientation.

### III. Structure de l'application :

**Fonction pour vérifier les fichiers photos/vidéos reçus  
(GlobalExceptionHandler.java) :**

```
@ControllerAdvice
public class GlobalExceptionHandler {
    private String maxSize;

    //StandardServletMultipartResolver
    @ExceptionHandler(MultipartException.class)
    public String handleErrorResponse1(MultipartException e, RedirectAttributes redirectAttributes) {
        redirectAttributes.addFlashAttribute("message", e.getMessage());
        return "redirect:/upload";
    }

    //CommonsMultipartResolver
    @ExceptionHandler(MaxUploadSizeExceededException.class)
    public String handleErrorResponse2(MaxUploadSizeExceededException e, RedirectAttributes redirectAttributes) {
        String message = "Erreur: taille maximum autorisée: " + maxSize;
        redirectAttributes.addFlashAttribute("message", message);
        return "redirect:/upload";
    }
}
```

Cette classe permet notamment de vérifier la taille des fichiers que l'utilisateur cherche à télécharger et renvoie une exception si elle est supérieur que la taille max défini dans Application.properties.

**Fonctions pour upload (FileUploadController.java) :**

```
@GetMapping("upload1")
public String listUploadedFiles(Model model) throws IOException {
    model.addAttribute("files", storageService.loadAll().map(
        path -> MvcUriComponentsBuilder.fromMethodName(FileUploadController.class,
            "serveFile", path.getFileName().toString()).build().toUri().toString()
    ).collect(Collectors.toList()));

    return "contribuer";
}

@GetMapping("/files/{filename:.+}")
@ResponseBody
public ResponseEntity<Resource> serveFile(@PathVariable String filename) {
    Resource file = storageService.loadAsResource(filename);
    return ResponseEntity.ok().header(HttpHeaders.CONTENT_DISPOSITION,
        "attachment; filename=\"" + file.getFilename() + "\"").body(file);
}
```

À partir de ces fonctions on a le chemin d'accès au fichier qui vont être upload dans l'espace de stockage prévu à cet effet.

```

@PostMapping("upload2")
public String handleFileUpload(@RequestParam(name="file") MultipartFile file, @RequestParam(name="id") Evenement event,
                               RedirectAttributes redirectAttributes, RedirectAttributes redirectInfo) {
    log.info("Téléchargement du fichier {}", file.getOriginalFilename());
    storageService.store(file);
    redirectAttributes.addFlashAttribute("message",
                                         "You successfully uploaded " + file.getOriginalFilename() + "!");
    Item newItem = new Item();
    newItem.nomItem = file.getOriginalFilename();
    newItem.typeItem = file.getContentType();
    List<Evenement> listEvenements = evenementRepository.findAll();

    for (Evenement e : listEvenements) {
        if (e.id == event.id) {
            newItem.setEvenement(e);
        }
    }

    String message;
    try {
        itemRepository.save(newItem);
    } catch (DataIntegrityViolationException e) {
        message = "Erreur : L'évènement '" + newItem.nomItem + "' existe déjà";
    }
    redirectInfo.addFlashAttribute("message", "You successfully created " + newItem.nomItem);

    return "redirect:/";
}

```

Cette fonction a pour but d'une part d'uploader le fichier de l'utilisateur et d'autre part de créer un nouvel Item avec les caractéristiques de ce fichier.

## Fonction pour enregistrer un nouvel événement (FormEventController.java) :

```

public String enregistrerNouvelEvenement(Evenement evenement, Model model, RedirectAttributes redirectInfo) {
    /**
     * Enregistre un nouvel evenement dans le repository et affiche un message de succès ou d'échec.
     * Les attributs sont donnés par l'utilisateur
     *
     * @param evenement le nouvel evenement créé dans le formulaire de contribuer.html à partir des données de l'utilisateur
     * @param model le model, utilisé dans les templates avec thymeleaf
     * @param redirectInfo les infos de redirection
     *
     * @return actualise la page contribuer.html et affiche le message de succès ou d'échec
     */
    String message;
    try {
        // cf. https://www.baeldung.com/spring-data-crud-repository-save
        evenementRepository.save(evenement);
        // Génération du message de succès
        message = "L'évènement '" + evenement.getNomEvenement() + "' a été correctement enregistré.";
    } catch (DataIntegrityViolationException e) {
        // Les noms sont définis comme 'UNIQUE'
        // En cas de doublon, JPA lève une exception de violation de contrainte d'intégrité
        message = "Erreur : L'évènement '" + evenement.getNomEvenement() + "' existe déjà";
    }
    // RedirectAttributes permet de transmettre des informations lors d'une redirection,
    // Ici on transmet un message de succès ou d'erreur
    // Ce message est accessible et affiché dans la vue 'contribuer.html'
    redirectInfo.addFlashAttribute("message", message);
}

return "redirect:/contribuer";
}

```

Cette fonction est appelée par le chemin "/contribuer/save" dans le template "contribuer.html". Elle prend en argument l'évènement créé à partir du formulaire à sauvegarder dans la base de données.

On peut voir ici que l'on sauvegarde l'évènement créé dans le dao avec la ligne evenementRepository.save(evenement). On renvoie un message de succès dans le cas où la sauvegarde réussie et un message d'erreur le cas échéant, comme par exemple si deux évènements ont le même nom.

## Fonction qui affiche un événement au hasard (IndexController.java) :

```

@GetMapping("eventRandom")
public String showEvenementRandom(Model model) {
    /**
     * Return la page wiki d'un évènement choisi au hasard
     *
     */

    List<Evenement> listEvenements = evenementRepository.findAll();
    Random random = new Random();
    int indexRandom = random.nextInt(listEvenements.size());
    Evenement evenementRandom = listEvenements.get(indexRandom);

    fillSsEventListOf(evenementRandom);
}

```

Cette fonction nous permet de rediriger vers la page wiki affiche un événement au hasard de la base de données. Elle utilise la fonction Random qui nous donne un entier aléatoire correspondant à un index de la liste d'événements. On va ensuite utiliser la fonction fillSsEventListOf() (qui sera détaillée plus tard) afin de récupérer la liste des sous-événements de l'événement choisi pour les afficher dans la liste du template "wiki.html".

```

// Vérification de la possibilité d'afficher la liste de sous évènement
boolean hasSsEvent = false;
if (!evenementRandom.getListeSousEvenements().isEmpty()) {
    hasSsEvent = true;

    ArrayList<LocalDate> days = new ArrayList<>();
    LocalDate dDebut;
    LocalDate dFin;

    for (Evenement e : evenementRandom.getListeSousEvenements()) {
        dDebut = e.getDateDebut();
        dFin = e.getDateFin();

        if (ChronoUnit.DAYS.between(dDebut, dFin) < 32) {
            days.add(dDebut);
            int i=0;
            while (i<days.size()-1) {
                if (!days.get(i).equals(days.get(i+1))) {
                    i+=1;
                } else {
                    days.remove(days.get(i+1));
                }
            }
        }
    }
}

```

Ici on récupère les jours des sous-événements et on supprime les dates doublons.

```

Collections.sort(days);

HashMap<LocalDate, List<String>> dateForDays = new HashMap<LocalDate, List<String>>();

for (LocalDate dDays : days) {
    List<String> eventForDays = new ArrayList<>();
    for (Evenement e : evenementRandom.getListeSousEvenements()) {
        if (dDays.equals(e.getDateDebut())) {
            eventForDays.add(e.getNomEvenement());
        }
    }
    dateForDays.put(dDays, eventForDays);
}

Map sortedDaysMap = new TreeMap(dateForDays);
model.addAttribute("days", days); //On ajoute la liste au modèle qui permet l'affichage
model.addAttribute("sortedDays", sortedDaysMap);
}

model.addAttribute("evenements", evenementRepository.findAll());
model.addAttribute("hasSsEvent", hasSsEvent);
model.addAttribute("evenement", evenementRandom);

```

Notre but ici pour finir l'affichage d'un événement est de créer des collections contenant les événements qui se passent un même jour. On intègre ensuite cette collection dans un dictionnaire qui va avoir une clé correspondant à l'élément commun de la date permettant de regrouper les événements afin de les afficher proprement dans la frise.

Le code model.addAttribute sert à attribuer la valeur à un élément qui se retrouvera dans la partie HTML.

### Fonction pour afficher les pages:

```

@GetMapping("contribuer")
public String showContribuer(Model model){
    /**
     * Montre la page pour ajouter un évènement
     */
    boolean fromWiki = false;
    model.addAttribute("fromWiki", fromWiki);
    model.addAttribute("evenement", new Evenement());

    return "contribuer";
}

```

### Explication fromWiki et utilisation

On a créé un boolean fromWiki pour savoir si l'information provient du controller de la page wiki ou de la page index (nous aurons sensiblement le même code pour le controller de Wiki). Ce booléen est ajouté au model permet à Thymeleaf de savoir s'il faut ou non afficher le nom et la valeur d'un événement principal. En effet, lorsque nous accédons à la page “contribuer.html” directement du header, l'événement que nous voulons créer n'a pas d'événement principal. Lorsque l'utilisateur arrive de la page d'un événement en cliquant sur “Ajouter un sous événement”, on veut ici définir au nouvel événement créé un événement principal.

```

@GetMapping("search")
public String showSearchPage(Model model) {
    return "search";
}

```

Ici on retourne le chemin qui va nous permettre d'aller à la page des recherches, c'est-à-dire "/search".

```

@GetMapping("ajoutSsEvent")
public String showContribuerAddSsEventTo(Model model, @RequestParam(name="id") Evenement evenement) {

    boolean fromWiki = true;
    model.addAttribute("fromWiki", fromWiki);
    model.addAttribute("evenement", new Evenement());
    model.addAttribute("evenementPrincipal", evenement);
    return "contribuer";
}

```

Ici on retourne le chemin qui va nous permettre d'aller à la page de contribution en indiquant que le futur événement créé sera un sous-événement de l'événement consulté sur le wiki.

### Fonction de remplissage des sous-événements (IndexController.java, ListEventController.java, SearchController.java, WikiController.java) :

```

public void fillSsEventListOf(Evenement event) {
    /**
     * Pour un événement donné, remplit la liste de sous-événements
     */
    List<Evenement> listeEvenement = evenementRepository.findAll();

    for (Evenement e:listeEvenement) {
        if ((event.getEvenementPrincipal() != null) && (e.getEvenementPrincipal() != null)) {
            if ((e.getEvenementPrincipal().getId() == event.getId()) && (e.getId() != event.getId())){
                if (!event.getListeSousEvenements().contains(e)){
                    event.getListeSousEvenements().add(e);
                }
            }
        }
    }
}

```

Le but de cette fonction est de remplir les listes de sous-événements des événements principaux. Pour cela, on crée une collection contenant tous les événements et on vérifie que les éléments à comparer n'aient pas d'événements principaux nuls. Ensuite nous vérifions si l'événement comparé possède le même idEventPrincipal que l'id de l'événement comparant sans que l'événement comparé soit le même que l'événement comparant.

## Fonction de la frise (IndexController.java, WikiController.java) :

```

@GetMapping("/")
public String showListEventPage(Model model) {
    /**
     * Author: léa
     * conception de la frise
     *
     */
    List<Evenement> events = evenementRepository.findAll(Sort.by(Sort.Direction.ASC, "dateDebut"));

    ArrayList<Integer> listeDatesDebutAnnee = new ArrayList<>();
    for (Evenement e : events) {
        listeDatesDebutAnnee.add(e.getDateDebut().getYear());
    }
    int i=0;
    while (i<listeDatesDebutAnnee.size()-1) {
        if (!listeDatesDebutAnnee.get(i).equals(listeDatesDebutAnnee.get(i+1))) {
            i+=1;
        } else {
            listeDatesDebutAnnee.remove(listeDatesDebutAnnee.get(i+1));
        }
    }

    HashMap<Integer, List<String>> da = new HashMap<Integer, List<String>>();

    for (Integer d: listeDatesDebutAnnee) {
        List<String> eventForD = new ArrayList<>();
        for (Evenement e : events) {
            if (d==e.getDateDebut().getYear()) {
                eventForD.add(e.getNomEvenement());
            }
        }
        da.put(d, eventForD);
    }
    Map sortedMap = new TreeMap(da);
    System.out.println(sortedMap);

    model.addAttribute("evenements", evenementRepository.findAll());
    model.addAttribute("dates", listeDatesDebutAnnee); //On ajoute la liste au modèle qui permet l'affichage
    model.addAttribute("da",sortedMap);
    return "index";
}

```

Cette frise a le même fonctionnement que celle explicité plus haut (voir l'explication de la fonction qui affiche un événement au hasard). A la différence de la première méthode, cette fonction ne prend en compte que l'année à laquelle l'événement a eu lieu. Cela permet de retracer entièrement tous les événements qui sont enregistrés sur le site par l'année à laquelle ils se sont produits.

## Fonction de suppression d'un événement (ListEventController.java):

```

@GetMapping(path = "delete")
public String supprimeUnEvenementPuisMontreLaListe(@RequestParam("id") Evenement evenement, RedirectAttributes redirectInfo) {
    /**
     * Fait appel à la fonction suppressionEventRecursive() de BoiteATools
     *
     */
    evenementRepository.delete(evenement);
    String message = "L'événement " + evenement.getNomEvenement() + " a bien été supprimé";

    // RedirectAttributes permet de transmettre des informations lors d'une redirection,
    // Ici on transmet un message de succès ou d'erreur
    // Ce message est accessible et affiché dans la vue 'liste_evenements.html'
    redirectInfo.addFlashAttribute("message", message);

    return "redirect:/wiki/liste_evenements"; // on se redirige vers l'affichage de la liste
}

```

Le but de cette fonction est de supprimer un événement dont on connaît l'id, on va récupérer ce dernier suite à l'appui d'un bouton sur une page HTML qui va permettre de récupérer l'id de l'événement concerné.

### Fonction d'affichage d'un événement choisi (ListEventController.java, SearchController.java, WikiController.java) :

```
@GetMapping(path = "showWikiPage")
public String showEventWikiPage(Model model, @RequestParam(name="id") Evenement evenement) {
```

Le code est à peu près similaire à la fonction qui ajoute un événement au hasard sauf qu'au lieu d'afficher un événement aléatoire on affiche un événement dont on connaît l'id (grâce au @RequestParam...).

### Fonction de remplissage des items (ListeRessourcesController.java, WikiController.java) :

```
public void fillItemListOf(Evenement evenement) {
    /**
     * Pour un évènement donné, rempli sa liste d'item
     */

    List<Item> liste_item = itemRepository.findAll();

    for (Item i: liste_item){
        if(i.getEvenement()==evenement){
            if(!evenement.getItems().contains(i)){
                evenement.getItems().add(i);
            }
        }
    }
}
```

Cette fonction remplit une collection d'items pour un événement dont l'id correspond avec l'id de l'événement attribué à l'item.

## Fonction d'affichage des items (ListRessourcesController.java, WikiController.java) :

```

@GetMapping("liste_item")
public String showListItemOf(Model model, @RequestParam(name="id") Evenement evenement) throws IOException{

    model.addAttribute("files", storageService.loadAll().map(
        path -> MvcUriComponentsBuilder.fromMethodName(FileUploadController.class,
            "serveFile", path.getFileName().toString()).build().toUri().toString())
    .collect(Collectors.toList()));

    fillItemListOf(evenement);
    List<Item> liste_item_of_event = evenement.getItems();
    List<String> liste_extensions = new ArrayList<>();
    boolean hasItems = false;
    if (!liste_item_of_event.isEmpty()){
        hasItems = true;
        for (Item i: liste_item_of_event){
            if (i.getTypeItem().contains("/")){
                liste_extensions.add(i.getTypeItem().split("/") [1]);
            }
            else{
                liste_extensions.add("undefined");
            }
        }
    }

    model.addAttribute("extensions", liste_extensions);
    model.addAttribute("hasItems", hasItems);
    model.addAttribute("evenement", evenement);
    model.addAttribute("liste_item_of_event", liste_item_of_event);

    return "liste_ressources_event";
}

```

Affichage des fichiers externes qui ont été uploadés et des fichiers appartenant déjà à la base de données.

## Fonction de vérification des mots recherchés (SearchController.java) :

```

private boolean motCleIsValidé(String motCle){
    /**
     * Vérifie si un mot clé est valide. Un mot clé est valide s'il contient AU MOINS un caractère de l'alphabet latin
     *
     * @param motCle Le mot clé à vérifier
     *
     * @return motCleIsValidé, boolean true si le mot clé est valide, false sinon
     */
    // On l'initialise à false
    boolean motCleIsValidé = false;

    // Tableau des caractères latin
    char[] charSearch = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'y'};
    // On parcourt les caractères du mot clé
    for(int i=0; i<motCle.length(); i++){
        {
            // On récupère le char à l'index i
            char chr = motCle.charAt(i);
            // On parcourt le tableau de caractère
            for(int j=0; j<charSearch.length; j++){
                {
                    // Si on trouve un caractère dans le mot clé, celui ci est valide
                    if(charSearch[j] == chr)
                    {
                        motCleIsValidé = true;
                    }
                }
            }
        }
    }

    return motCleIsValidé;
}

```

Le but de cette fonction est de vérifier si un mot est valide, on a décidé qu'à partir du moment où un caractère était issu de l'alphabet latin, le mot était valide.

## Fonction de recherche par mot-clé(SearchController.java) :

```

@GetMapping("fctSearch")
private String fctSearchByMotCle(Model model, @RequestParam(name="motCle", defaultValue "") String motCle){

    boolean recherche_en_cours = true; // Pour indiquer à thymeleaf d'avoir une recherche en cours
    boolean recherche_par_mot_cle = true; // Pour indiquer à thymeleaf d'avoir une recherche par mot clé
    boolean motCleIsValide = motCleIsValide(motCle); // Vérification de la validité du mot clé
    boolean resultats_trouves = false; // Indiquer à thymeleaf si des résultats ont été trouvés

    // On ajoute tous les booléens au modèle afin de pouvoir les utiliser
    model.addAttribute("recherche_en_cours", recherche_en_cours);
    model.addAttribute("recherche_par_mot_cle", recherche_par_mot_cle);
    model.addAttribute("isValid", motCleIsValide);
    model.addAttribute("motCle", motCle);

    if (motCleIsValide) {

        List<Evenement> listEvenementSearched = evenementRepository.findByNomEvenementContains(motCle);
        //On vérifie aussi si le lieu contient le mot clé
        List<Evenement> listEvenementSearchedByLieu = evenementRepository.findByLieuContains(motCle);

        if (!listEvenementSearchedByLieu.isEmpty()){
            for (int i=0; i<listEvenementSearchedByLieu.size(); i++){
                if (!listEvenementSearched.contains(listEvenementSearchedByLieu.get(i))){
                    //On ajoute les résultats de la recherche par lieu à la liste des événements déjà trouvés
                    listEvenementSearched.add(listEvenementSearchedByLieu.get(i));
                }
            }
        }
        //Si la liste de résultat n'est pas vide, c'est qu'on a trouvé des résultats
        if (!listEvenementSearched.isEmpty()){
            resultats_trouves = true;
        }
        //Ajout du booléen au modèle
        model.addAttribute("results", listEvenementSearched);
    }

    //Ajout de la liste de résultat
    model.addAttribute("trouves", resultats_trouves);

    //renvoie vers search.html
    return "search";
}

```

Le but de cette fonction est de réaliser la recherche par mot clé, pour cela, on vérifie que le mot recherché est valide, on le compare ensuite aux noms des événements et aux lieux pour savoir s'il est dans la base de données, s'il l'est on le récupère et on récupère un booléen signifiant qu'on a reçu le résultat.

## Fonction de recherche par date -- à implémenter -- (SearchController.java) :

```

private String fctSearchByDate(Model model,
    @RequestParam(name="month", defaultValue = "") String month,
    @RequestParam(name="year", defaultValue = "") String year,
    @RequestParam(name="day", defaultValue = "") String day){

    /**
     * Methode pour la recherche par date
     *
     * TODO
     */

    boolean recherche_par_date = true;
    model.addAttribute("recherche_par_date", recherche_par_date);
    model.addAttribute("year", year);

    return "search";
}

```

Le but de cette fonction est de faire une recherche par date cependant, elle n'est pas complète et ce sera au prochain groupe d'implémenter cette dernière.

## Fonction d'affichage/cache des événements liés à une date (index.js):

```

let dates = document.getElementsByClassName("date");
let listeEvenement = document.getElementsByClassName("liste-evenement");

for(let i = 0; i< dates.length;i++){
    dates[i].addEventListener('click',affichage);
}

function searchIndexOf(lienCible){
    let index = 0;
    if (lienCible.className == "date"){
        for (let i=0; i<dates.length;i++){
            return index;
        }
    }
}

function affichage(event){
    let lienCible = event.target;
    let index = searchIndexOf(lienCible);
    let listeUtile = listeEvenement[index];
    if (getComputedStyle(listeUtile).display != "none") {
        listeUtile.style.display = "none";
    } else {
        listeUtile.style.display = "block";
    }
}

```

Le but de la fonction searchIndexOf() est de trouver l'indice du bouton cliqué parmi tous les boutons "dates" existants, la fonction affichage() vérifie si l'affichage des événements est actif ou non et désactive ou active l'affichage en fonction.

## Fonction d'initialisation de la carte Google Maps (map.js):

```
function initMap() {
    map = new google.maps.Map(document.getElementById("map"), {
        center: { lat: 43.624227611324216, lng: 2.268580304110914 },
        zoom: 15,
    });
}
```

Le but de cette fonction est de placer la carte, par défaut, à un point choisi arbitrairement (l'école d'ingénieur ISIS).

## Fonction de placement de la carte (map.js):

```
var adresse = [];
initMap();
$.ajax({
    url : url + "/api/evenements"
}).done(function(dataJSON) {
    $.ajax({
        url : url + "/api/evenements/" + dataJSON.content[0].id + "/items"
    }).done(function() {
        for (let j=0; j<dataJSON.content.length; j++) {
            adresse[j] = dataJSON.content[j].lieu;
        }
        for (let k=0; k<adresse.length; k++) {
            if (adresse[k] == document.getElementById("adresse").attributes.value.textContent) {
                lieu = adresse[k];
                const geocoder = new google.maps.Geocoder();
                geocoder.geocode({ 'address': lieu }, function(results, status) {
                    if (status == 'OK') {
                        map.setCenter(results[0].geometry.location);
                    }
                });
            }
        }
    });
});
```

Le but de cette première partie de fonction est de récupérer le lieu d'un événement et d'utiliser un géocodage pour en faire la recherche et l'afficher sur la carte.

```

$.ajax({
    url : url + "/items"
}).done(function(dataJson) {
    for (let i=0; i<dataJson.content.length; i++) {
        $.ajax({
            url : url+"/items/"+dataJson.content[i].id+"/position"
        })
        .done(function(json) {
            var marker = new google.maps.Marker({
                position: {lat: parseFloat(json.latitude), lng: parseFloat(json.longitude)},
                title: dataJson.content[i].nomItem,
                map: map
            });
            var infowindow = new google.maps.InfoWindow({
                content: "<p>" + dataJson.content[i].nomItem + "</p>"
            });
            marker.addListener("click", () => {
                infowindow.open(map, marker);
            });
        });
    }
});
}

```

Le but de la deuxième partie de la fonction est de récupérer les items composants un événement et de placer des marqueurs sur leurs localisations.

### Avancée de la fonction de recherche des dates -- à implémenter-- (search.js):

```

function populateDays(month) {
    // On supprime les éléments <option> pour l'élément
    // <select> des jours afin de pouvoir ajouter les prochains
    while(daySelect.firstChild) {
        daySelect.removeChild(daySelect.firstChild);
    }

    // On crée une variable afin de contenir le nombre
    // de jours à afficher
    var dayNum;

    // 31 ou 30 jours ?
    if(month === 'Janvier' || month === 'Mars' || month === 'Mai' || month === 'Juillet' || month === 'Août' || month === 'Octobre' || month === 'Décembre') {
        dayNum = 31;
    } else if(month === 'Avril' || month === 'Juin' || month === 'Septembre' || month === 'Novembre') {
        dayNum = 30;
    } else {
        // Si le mois est février, on calcule si l'année est bissextile
        var year = yearSelect.value;
        var leap = new Date(year, 1, 29).getMonth() === 1;
        dayNum = leap ? 29 : 28;
    }
}

```

```

// on ajoute le bon nombre de jours dans autant
// d'éléments <option> pour l'élément <select>
// pour la journée
let no_choice_day = document.createElement('option');
no_choice_day.textContent = "-- Jour --";
no_choice_day.value = "";
daySelect.appendChild(no_choice_day);

for(i = 1; i <= dayNum; i++) {
    var option = document.createElement('option');
    option.textContent = i;
    option.name="day";
    daySelect.appendChild(option);
}

// Si le jour précédent a déjà été défini on utilise
// la valeur de ce jour pour daySelect afin d'éviter de
// réinitialiser le jour lorsqu'on change l'année
if(previousDay) {
    daySelect.value = previousDay;

    // Si le jour précédent correspond au dernier jour d'un mois
    // et que le mois sélectionné possède moins de jours (par
    // exemple en février)
    if(daySelect.value === "") {
        daySelect.value = previousDay - 1;
    }

    if(daySelect.value === "") {
        daySelect.value = previousDay - 2;
    }

    if(daySelect.value === "") {
        daySelect.value = previousDay - 3;
    }
}
}

```

Le but est de récupérer le bon nombre de jours à proposer à l'utilisateur en fonction du mois sélectionné.

```

] function populateYears() {
    // On obtient l'année courante
    var date = new Date();
    var year = date.getFullYear();
    // On affiche l'année courante et les 100 années
    // précédentes pour l'élément <select> destiné à
    // stocker l'année
] for(var i = 0; i <= 100; i++) {
    var option = document.createElement('option');
    option.textContent = year-i;
    option.name="year";
    yearSelect.appendChild(option);
}

// Lorsque la valeur du mois ou de l'année est modifiée
// on relance populateDays()
] yearSelect.onchange = function() {
    populateDays(monthSelect.value);
}

] monthSelect.onchange = function() {
    populateDays(monthSelect.value);
}

// On conserve le jour sélectionné
var previousDay;
] // On met à jour la journée utilisé précédemment
// (voir la fin de populateDays() pour voir où
// est utilisée cette valeur)
] daySelect.onchange = function() {
    previousDay = daySelect.value;
}

```

Code commenté, le but est de récupérer l'année courante. Je me suis inspiré du site pour implémenter cette fonction

<https://developer.mozilla.org/fr/docs/Web/HTML/Element/Input/date>

## Fonction d'affichage des modificateurs (liste\_events.js):

L'objectif de ce script est de déterminer quel évènement l'utilisateur souhaite modifier ou supprimer. Pour cela, il récupère les indices des boutons cliqués, et transforme les bonnes cases en input.

```

function afficher_modificateur(event) {
    console.log("clickme");
    index_btn = searchIndexOf(event.target);
    console.log(index_btn);

    if (event.target.className == "btn_modif") {
        console.log("on est là");
        let btnCible = event.target;
        btnCible.style.display = "none";
        btns_retour[index_btn].style.display = "block";
        btns_save[index_btn].style.display = "block";
        liste_espaces_modif[index_btn].style.display = "block";

        console.log(liste_dateDebut[index_btn]);
        console.log(manipulationDate(liste_dateDebut[index_btn]));
        nomEvenementToInput(liste_nomEvenementCells[index_btn], liste_nomEvenements[index_btn]);
        lieuToInput(liste_lieuCells[index_btn], liste_lieux[index_btn]);
        descriptionToTextArea(liste_descriptionCells[index_btn], liste_descriptions[index_btn]);
        dateDebutToInput(liste_dateDebutCells[index_btn], liste_dateDebut[index_btn]);
        dateFinToInput(liste_dateFinCells[index_btn], liste_dateFin[index_btn]);
    }

    else if(event.target.className == "btn_retour"){
        window.location.reload();
        let btnCible = event.target;
        btnCible.style.display = "none";
        btns_save[index_btn].style.display = "none";
        liste_espaces_modif[index_btn].style.display = "none";
        btns_modif[index_btn].style.display = "block";

        inputToNomEvenement(liste_nomEvenementCells[index_btn], liste_nomEvenements[index_btn]);
        inputToLieu(liste_lieuCells[index_btn], liste_lieux[index_btn]);
        textAreaToDescription(liste_descriptionCells[index_btn], liste_descriptions[index_btn]);
        inputToDateDebut(liste_dateDebutCells[index_btn], liste_dateDebut[index_btn]);
        inputToDateFin(liste_dateFinCells[index_btn], liste_dateFin[index_btn]);
    }

    else if(event.target.className == "btn_save") {

        let btnCible = event.target;
        btnCible.style.display = "none";
        btns_modif[index_btn].style.display = "block";
        btns_retour[index_btn].style.display = "none";

        liste_nomEvenements[index_btn] = liste_nomEvenementCells[index_btn].firstChild.value;
        liste_lieux[index_btn] = liste_lieuCells[index_btn].firstChild.value;
        liste_descriptions[index_btn] = liste_descriptionCells[index_btn].firstChild.value;
        liste_dateDebut[index_btn] = liste_dateDebutCells[index_btn].firstChild.value;
        liste_dateFin[index_btn] = liste_dateFinCells[index_btn].firstChild.value;

        let evenement = {
            id: liste_id[index_btn],
            nomEvenement: liste_nomEvenements[index_btn],
            description: liste_descriptions[index_btn],
            lieu: liste_lieux[index_btn],
            dateDebut: liste_dateDebut[index_btn],
            dateFin: liste_dateFin[index_btn]
        }
    }
}

```

```

        console.log(événement);
        $.ajax({
            type: 'PUT',
            url: url+ "/" + événement.id,
            contentType: 'application/json',
            data: JSON.stringify(événement), // access in body
        }).done(function () {
            console.log('SUCCESS');
            window.alert("L'évènement a été modifié!");
            window.location.reload();
        }).fail(function (msg) {
            console.log('FAIL');
        }).always(function (msg) {
            console.log('ALWAYS');
        });
    });
}

```

Enfin, cette fonction finit par une requête ajax qui modifie l'événement dans l'api. L'api est accessible à l'url localhost:8080/api en local ou <https://lit-plains-26980.herokuapp.com/api> pour l'application déployée.

### Fonction de recherche d'index (list\_event.js):

```

function searchIndexOf(btnCible) {
    let index = 0;
    if (btnCible.className == "btn_modif") {
        for (let i=0; i<btms_modif.length;i++) {
            if (btnCible == btms_modif[i]){
                return index;
            }
            else{
                index+=1;
            }
        }
    }
}

```

Le but est de récupérer l'indice du bouton cliqué, même principe pour les boutons de modifications (ici à l'écran), de sauvegarde des modifications et des suppressions.

### Fonctions de modifications des événements (list\_event.js):

```

/// MODIFICATION EVENEMENT

function nomEvenementToInput(nomEvenementCell, nomEvenement) {

    nomEvenementCell.textContent = "";
    nomEvenementCell.innerHTML = "<input type=\"text\" value=\"" + nomEvenement + "\"></input>";
}

function inputToNomEvenement(nomEvenementCell, nomEvenement) {

    nomEvenementCell.innerHTML = "";
    nomEvenementCell.textContent = nomEvenement;
}

```

Remplacement des attributs de l'événement choisi. Format identique pour les autres attributs de l'événement.

### Fonction de récupération des dates (liste\_event.js):

```
function manipulationDate(date) {
    /**
     *
     * Retourne une date au format yyyy-mm-dd
     */
    let day_month_year = date.split("/");
    return day_month_year[2] + "-" + day_month_year[1] + "-" + day_month_year[0];
}
```

Change le format de la date.

### Fonction de récupération des attributs par l'id des boutons (liste\_event.js):

```
// Modification effective

function findIdFromBtn(btnCible) {
    let index = searchIndexOf(btnCible);
    return liste_id[index];
}
```

Récupère l'id des boutons et va chercher l'attribut de l'événement correspondant à l'id du bouton. Même format pour les autres attributs d'un événement.

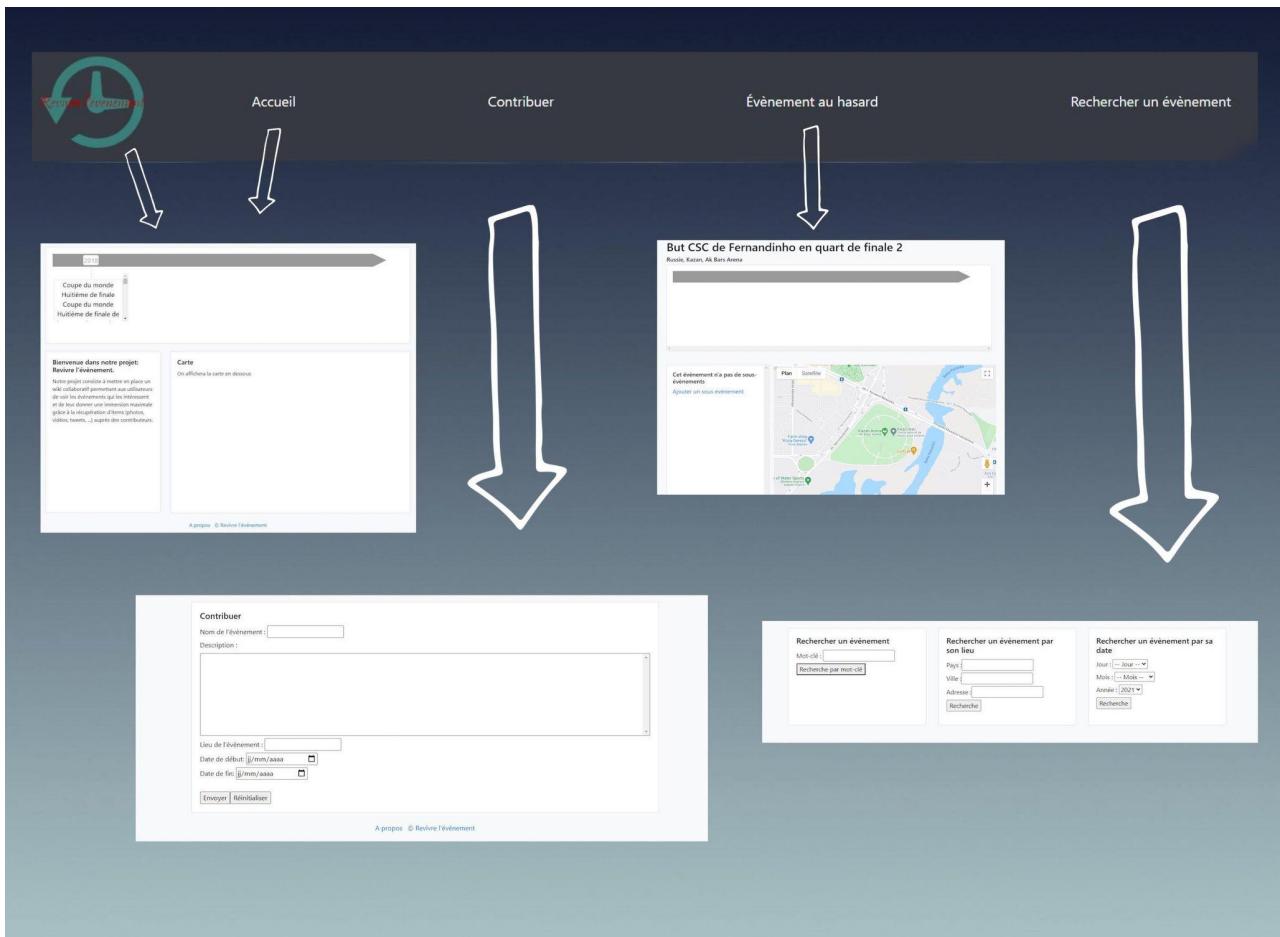
### Fonction de confirmation de la suppression (liste\_event.js):

```
function confirmation_suppr(event) {
    console.log(liste_suppr);
    let index_btn = searchIndexOf(event.target);
    event.target.style.display = "none";
    console.log(index_btn);
    window.alert("Veuillez cliquer une deuxième fois!");
    liste_suppr[index_btn].style.display = "block";
}
```

Le but de la fonction est de faire réagir l'utilisateur s'il a fait une erreur au moment d'effectuer une suppression. La fonction demande confirmation de la suppression en faisant apparaître une notification confirmant la demande de suppression, suite à cela, il y a l'apparition du bouton qui permet la suppression des événements.

## Lien entre les différentes pages de l'application web

Les pages accessibles depuis le Header :



À partir du Header on peut retourner sur la page index (l'accueil du site) en cliquant sur le logo ou Accueil.

On peut aussi accéder à la page Contribuer qui permet d'ajouter un événement.

Cliquer sur Evenement au hasard depuis le header permet d'accéder à la page d'un événement déjà enregistré dans la base de donné et pris au hasard.

Enfin le header peut nous rediriger vers la page Recherche un événement à partir de laquelle on peut chercher un événement précis que l'on souhaite afficher.

Les pages accessibles depuis la page d'accueil ou depuis la page d'un événement :

Bienvenue dans notre projet:  
Révisez l'événement.

Notre projet consiste à mettre en place un calendrier d'événements sportifs et culturels et à vous permettre de voir les événements qui les intéressent et de leur donner une visibilité maximale grâce à la réception d'informations, visuels, tweets, ... des contributeurs.

**Carte**  
On affichera la carte en dessous.

A propos | © Révisez l'événement

**Description**  
But CSC Fennmarken à 11 min de jeu : Sur le corner frappé à gauche par Nacer Chadi, Vincent Kompany efface le ballon de la tête au premier poteau et saupoudre l'arbitrage, qui frappe son gardien de près avec l'épaule droite.

Liste des événements | Liste des ressources | Fichier à télécharger | Choisir un fichier | Aucun fichier choisi | Upload | Télécharger |

A propos | © Révisez l'événement

**Liste**  
L'événement Victoire de la France a bien été supprimé

Nom de l'événement	Description	Lieu	Date de début	Date de fin	Page web
La Coupe du monde de football 2018 est la 21 <sup>e</sup> édition de la compétition de football organisée par la FIFA qui réunit les trente-deux sélections nationales masculines issues de la phase qualificative. Russe du 14 juin au 15 juillet 2018 et est remportée par l'équipe de France.	Russie	14/06/2018	15/07/2018		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Éditer</a>
But d'Casan à 42 min de jeu : La Grèce repousse l'avantage au bout d'un contre-attaque lancé par son gardien. La défense de la Grèce est défaillante et laisse un espace à l'attaquant Casan. Dans la surface, le Français frappe sans contrôle du droit du pied mais ne va pas au but.	Russie, Stade Fédéral Olympique	30/06/2018	30/06/2018		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Éditer</a>
But de Kaban à 40 min de jeu : L'Algérie repousse l'avantage au bout d'un contre-attaque lancé par son gardien. La défense de l'Algérie est défaillante et laisse un espace à l'attaquant Kaban. Dans la surface, le Français frappe sans contrôle du droit du pied mais ne va pas au but.	Russie, Stade Fédéral Olympique	30/06/2018	30/06/2018		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Éditer</a>
France - Espagne : Finale - Résultat	Russie, Ak Bars Arena	30/06/2018	30/06/2018		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Éditer</a>
France - Espagne : A-3	Russie, Ak Bars Arena	30/06/2018	30/06/2018		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Éditer</a>
But A. Glikmanas à 11 min de jeu : Sur penalty, le Français frappe sur la gauche et prend Arnaud à contre pied.	Russie, Stade Fédéral Olympique	30/06/2018	30/06/2018		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Éditer</a>
But A. Di Maria à 41 min de jeu : Suite à une touche côté gauche, Di Maria hérite du ballon à 25 mètres dans l'axe. Il décale une incisive passe à droite pour que le latéral ait une chance à 10m.	Russie, Krasnodar Arena	30/06/2018	30/06/2018		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Éditer</a>
But de Mercado en huitième de finale de la coupe du monde 2	Russie, Krasnodar Arena	30/06/2018	30/06/2018		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Éditer</a>
But G. Merkalo à 48 min de jeu : Sur le coup franc, Merkalo décale une passe à droite pour que le latéral ait une chance à 10m.	Russie, Krasnodar Arena	30/06/2018	30/06/2018		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Éditer</a>

0 1 2 3 4 5 6

A propos | © Révisez l'événement

**Liste**  
L'événement Victoire de la France a bien été supprimé

Nom de l'événement	Description	Lieu	Date de début	Date de fin	Page web
La Coupe du monde de football 2018 est la 21 <sup>e</sup> édition de la compétition de football organisée par la FIFA qui réunit les trente-deux sélections nationales masculines issues de la phase qualificative. Russe du 14 juin au 15 juillet 2018 et est remportée par l'équipe de France.	Russie	14/06/2018	15/07/2018		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Éditer</a>
But d'Casan à 42 min de jeu : La Grèce repousse l'avantage au bout d'un contre-attaque lancé par son gardien. La défense de la Grèce est défaillante et laisse un espace à l'attaquant Casan. Dans la surface, le Français frappe sans contrôle du droit du pied mais ne va pas au but.	Russie, Stade Fédéral Olympique	30/06/2018	30/06/2018		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Éditer</a>
But de Kaban à 40 min de jeu : L'Algérie repousse l'avantage au bout d'un contre-attaque lancé par son gardien. La défense de l'Algérie est défaillante et laisse un espace à l'attaquant Kaban. Dans la surface, le Français frappe sans contrôle du droit du pied mais ne va pas au but.	Russie, Stade Fédéral Olympique	30/06/2018	30/06/2018		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Éditer</a>
France - Espagne : Finale - Résultat	Russie, Ak Bars Arena	30/06/2018	30/06/2018		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Éditer</a>
France - Espagne : A-3	Russie, Ak Bars Arena	30/06/2018	30/06/2018		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Éditer</a>
But A. Glikmanas à 11 min de jeu : Sur penalty, le Français frappe sur la gauche et prend Arnaud à contre pied.	Russie, Stade Fédéral Olympique	30/06/2018	30/06/2018		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Éditer</a>
But A. Di Maria à 41 min de jeu : Suite à une touche côté gauche, Di Maria hérite du ballon à 25 mètres dans l'axe. Il décale une passe à droite pour que le latéral ait une chance à 10m.	Russie, Krasnodar Arena	30/06/2018	30/06/2018		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Éditer</a>
But de Mercado en huitième de finale de la coupe du monde 2	Russie, Krasnodar Arena	30/06/2018	30/06/2018		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Éditer</a>
But G. Merkalo à 48 min de jeu : Sur le coup franc, Merkalo décale une passe à droite pour que le latéral ait une chance à 10m.	Russie, Krasnodar Arena	30/06/2018	30/06/2018		<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Éditer</a>

A propos | © Révisez l'événement

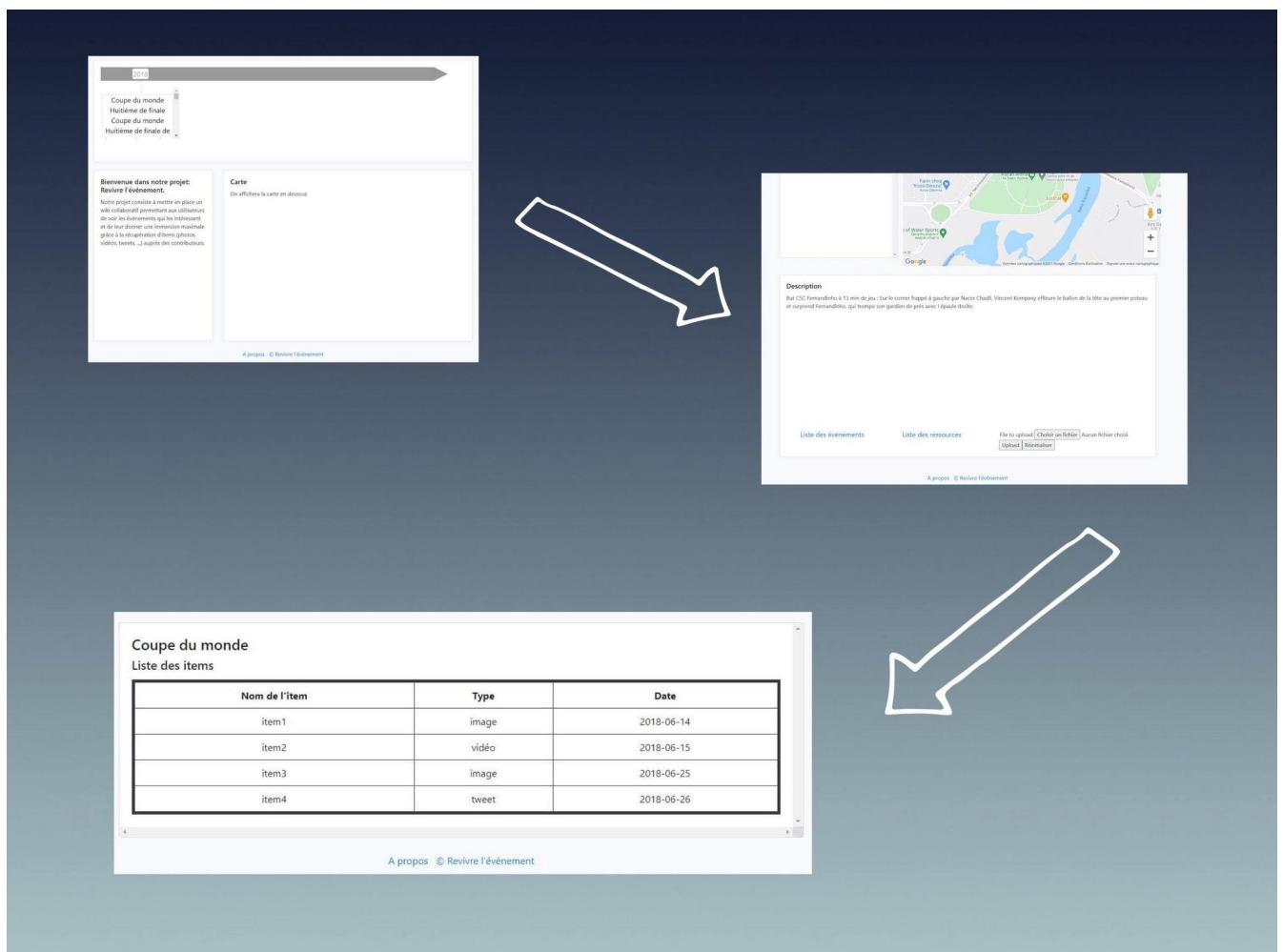
**Description**  
But CSC Fennmarken à 11 min de jeu : Sur le corner frappé à gauche par Nacer Chadi, Vincent Kompany efface le ballon de la tête au premier poteau et saupoudre l'arbitrage, qui frappe son gardien de près avec l'épaule droite.

Liste des événements | Liste des ressources | Fichier à télécharger | Choisir un fichier | Aucun fichier choisi | Upload | Télécharger |

A propos | © Révisez l'événement

Depuis la frise affichée sur la page d'accueil on a accès aux pages de tous les événements enregistrés dans la base de donnée et classés par an.

Une fois sur la page de l'événement on peut accéder à la Liste des événements enregistrés au niveau de laquelle on peut modifier ou supprimer n'importe lequel de ces événements.



The screenshot displays a web application interface for managing events. At the top, there's a timeline bar with the year 2018 and several event markers: 'Coupe du monde', 'Huitième de finale', 'Coupe du monde', and 'Huitième de finale'. Below the timeline, there are two main sections: 'Bienvenue dans notre projet: Revivre l'événement.' and 'Carte' (Map). The 'Carte' section includes a map of a stadium area with various landmarks labeled. A large white arrow points from the 'Carte' section towards the right side of the screen. On the right side, there's a detailed view of an event description for a football match at CSC Femandroitsa. The description text is as follows:

**Description**

Bat CSC Femandroitsa à 33 min de jeu : Sur le corner, frappé à gauche par Nacer Chabli, Vincent Kompany efface le ballon de la tête au premier poteau et signifie l'exploit, qui échappe aux gardiens de près avec l'option droite.

Below the description, there are buttons for 'Liste des événements' and 'Liste des ressources'. Further down, there are upload buttons: 'Fichier à upload' (File to upload), 'Choisir un fichier' (Select file), and 'Annuler' (Cancel). Another large white arrow points from the bottom left towards the bottom center of the screen. In the bottom center, there's a modal window titled 'Coupe du monde' with the sub-section 'Liste des items'. This modal contains a table with four items:

Nom de l'item	Type	Date
item1	image	2018-06-14
item2	vidéo	2018-06-15
item3	image	2018-06-25
item4	tweet	2018-06-26

At the bottom of the modal, there are links for 'A propos' and 'Revivre l'événement'.

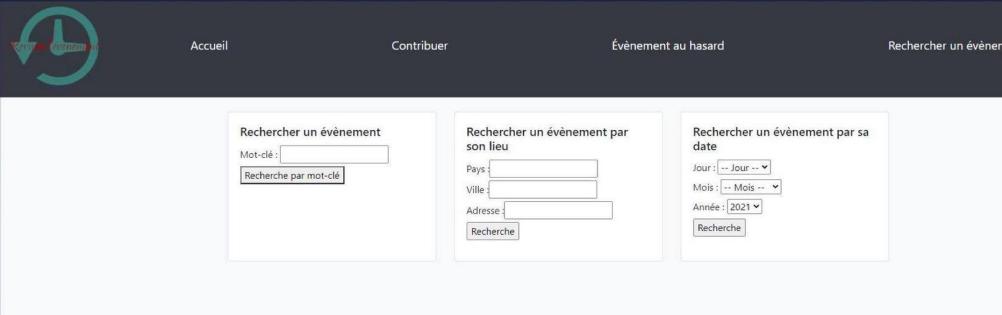
À partir de cette même page de l'événement en question on peut avoir accès à la liste des items en lien avec cet événement et qui ont déjà été upload.

Il est aussi possible d'ajouter de nouveaux items (c'est-à-dire des audios, des vidéos, des photos avec une taille max de 100 mb valeur modifiable dans application properties).

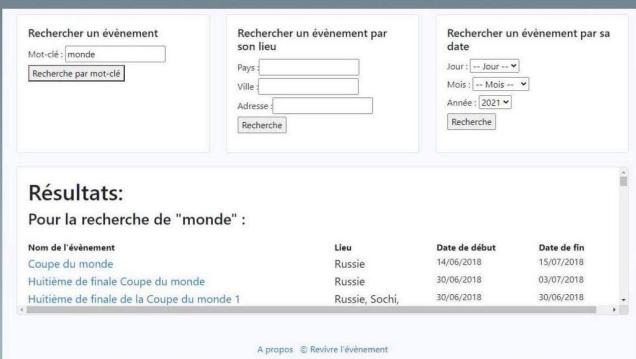
The screenshot shows two versions of a 'Contribuer' (Contribute) form. The top version is the initial form with empty fields. The bottom version shows the same form after an event has been successfully registered, with a confirmation message: 'L'évènement 'Victoire de la France' a été correctement enregistré.' (The event 'Victoire de la France' has been correctly registered.)

Une fois sur la page Contribuer, on peut rentrer les informations nécessaires à la génération d'un nouvel événement (son nom, quelques lignes le décrivant, la date où il commence et la date où il termine)

Une fois enregistré, on peut retrouver cet événement sur la frise en page d'accueil ou dans le tableau de la page Liste événement (ou encore en cliquant sur Evenement au hasard si vous avez de la chance)



A large white downward arrow points from the top search interface to the results page below.



Nom de l'événement	Lieu	Date de début	Date de fin
Coupe du monde	Russie	14/06/2018	15/07/2018
Huitième de finale Coupe du monde	Russie	30/06/2018	03/07/2018
Huitième de finale de la Coupe du monde 1	Russie, Sochi,	30/06/2018	30/06/2018

Une fois sur la page Rechercher un événement, il nous est possible de retrouver un événement en particulier soit en filtrant par un mot clé, un lieu ou une date.

## IV.Points d'améliorations

Par manque de temps, nous n'avons pas pu rendre l'application aussi complète que ce que nous aurions voulu. Tout d'abord et l'un des points les plus importants à faire, est de réussir à correctement ajouter les sous-événements. En effet, lors de la création de sous-événements, nous souhaitons assigner le futur élément créé à la collection de son événement principal. Malheureusement, cela ne marche pas et nous narrivons qu'à créer de nouveaux événements indépendants et nous n'avons pas trouvé comment résoudre ce problème.

Actuellement, toute personne arrivant sur le site est capable de modifier ou supprimer des événements, il faudrait mettre en place un système d'authentification pour n'autoriser que certaines personnes à accéder à ces fonctionnalités.

De plus, il est possible d'améliorer notre projet en implémentant la fonction de recherche par date, dans le cas où l'utilisateur ne connaisse pas de mot-clé en rapport avec l'événement qu'il cherche ou encore le lieu de ce dernier, ou encore par simple curiosité.

En ce qui concerne les recherches, il faudrait aussi implémenter celle en rapport avec les lieux, la recherche par lieu peut déjà être faite à l'aide de la recherche par mot-clé, mais pour des raisons de clarté il serait préférable de différencier les deux. Il faudrait également rendre la recherche par mot-clé insensible à la casse.

En complément de cela, il faudrait pouvoir permettre à l'utilisateur de renseigner la position des items qu'il souhaite rajouter dans la base de données afin d'avoir des positions précises des items et donc de pouvoir connaître la surface réelle de l'événement mais aussi la surface de réaction engendrée par un événement (on pourrait estimer qu'un match de foot a une surface réelle de quelques kilomètres carrés contenant des événements comme les buts et donc les items liés à ce dernier telle qu'une vidéo amateur du but, mais que ce match aurait une surface de réaction de plusieurs centaines de kilomètres carrés si des personnes réagissent sur Twitter à propos de ce match aux quatre coins de la France).

De la même façon, il faudrait pouvoir permettre au contributeur de renseigner l'orientation de sa photo ou vidéo et l'utiliser sur Google Maps pour ajouter de l'immersion à l'utilisateur lambda et lui permettre de revivre l'événement dans des conditions optimales.

Pour finir, il reste des détails au niveau du front-end à améliorer et la navigation entre les pages webs et ces dernières elles-mêmes sont à optimiser, notamment la page wiki dans laquelle les boutons de la frise ne permettent pas d'afficher ou non la liste des événements d'une année en fonction de l'envie de l'utilisateur et un problème au niveau de l'affichage responsive:

- les dates de la frise se superposent quand l'écran devient trop petit et que l'on a au moins quatre sous-événements;
- la carte issue de google Maps ne s'affiche pas pour un écran ayant une épaisseur inférieure à 992px;