```
const Toast = Swal.mixin({
  toast: true,
  position: 'top-end',
  showConfirmButton: false,
  timer: 1500,
  timerProgressBar: true
});

var pageName = window.location.pathname.substring(
  location.pathname.lastIndexOf("/") + 1
);

$(document).ready(() => {
  switch (pageName) {
    case "index.php":
    case "":
      GetPublicEvents();
      break;
    case "createEvent.php":
      $("#beginningDate").focus();
      $("#beginningTime").focus();
      $("#name").focus();
      break;
    case 'event.php':
      let urlParams = new URLSearchParams(window.location.search);
      let idEvent = urlParams.get('idEvent');

      GetEventData(idEvent)
      break;
    case 'manageEvents.php':
      GetManageEvents();
      break;
    case 'editEvent.php':
      $("#nbMaxGuest").focus();
      $("#eventDescription").focus();
      $("#beginningDate").focus();
      $("#beginningTime").focus();
      $("#endDate").focus();
      $("#endTime").focus();
      $("#place").focus();
      $("#eventName").focus();
      break;
  }

  $("#btnCreateEvent").click(CreateEvent);
});

// Change de couleur l'icone de l'image lorsqu'une image est sélectionné dans la
création ou la modification de l'événement
$("#img").on('change', () => {
  $("#img")[0].parentNode.parentNode.children[0].style.color = "#38d39f";
});

/**
```

```
 * @author Hoarau Nicolas
 * @date 27.05.20
 * @brief Fonction qui envoie les données pour créer un événement via un call ajax
 * @param {*} event
 * @version 1.0
 */
async function CreateEvent(event) {
  if (event)
    event.preventDefault();

  //#region Initialisation
  let formdata = new FormData();
  let eventName = $("#eventName").val();
  let place = $("#place").val();
  let eventDescription = $("#eventDescription").val();
  let nbMaxGuest = $("#nbMaxGuest").val();
  let type = $('input[name="type"]:checked').val();
  let img = document.getElementById("img").files[0];
  let beginningDate = new Date($("#beginningDate").val());
  let endDate = new Date($("#endDate").val());
  let beginningTime = $("#beginningTime").val();
  let endTime = $("#endTime").val();
  let now = new Date();

  beginningDate.setHours(beginningTime.split(':')[0], beginningTime.split(':')[1], 0,
0);
  endDate.setHours(endTime.split(':')[0], endTime.split(':')[1], 0, 0);
  now.setHours(now.getHours(), now.getMinutes(), 0, 0);
  //#endregion

  //#region Conditions
  if (eventName.length == 0) {
    $("#eventName").focus();
    return;
  } else {
    formdata.append("eventName", eventName);
  }

  if (eventDescription.length == 0) {
    $("#eventDescription").focus();
    return;
  } else {
    formdata.append("eventDescription", eventDescription);
  }

  if (place.length == 0) {
    $("#place").focus();
    return;
  } else {
    formdata.append("place", place);
  }

  if (nbMaxGuest == 0 || nbMaxGuest == "") {
    $("#nbMaxGuest").focus();
    return;
```

```
108   } else {
109     formdata.append("nbMaxGuest", nbMaxGuest);
110   }
111
112   if (beginningDate === null) {
113     $("#beginningDate").focus();
114     return;
115   } else if (beginningDate < now) {
116     Swal.fire({
117       title: "Création de l'événement",
118       text: "La date choisis est déjà passé",
119       icon: "error",
120       timer: 1500,
121       showConfirmButton: false
122     });
123     return;
124   } else {
125
126     formdata.append("beginningDate", beginningDate.toLocaleDateString());
127     formdata.append("beginningTime", beginningTime)
128   }
129
130   if (endDate === null) {
131     $("#endDate").focus();
132     return;
133   } else if (endDate < beginningDate) {
134     Swal.fire({
135       title: "Création de l'événement",
136       text: "La date choisis est avant le début de l'événement",
137       icon: "error",
138       timer: 1500,
139       showConfirmButton: false
140     });
141     return;
142   } else {
143     formdata.append("endDate", endDate.toLocaleDateString());
144     formdata.append("endTime", endTime);
145   }
146
147   if (typeof img !== 'undefined') {
148     formdata.append("img", img);
149   }
150
151   if (typeof type !== 'undefined') {
152     if (type == 'public') {
153       formdata.append("type", 0);
154     } else {
155       formdata.append("type", 1);
156       formdata.append('guestlist', await GetGuestList());
157     }
158   } else {
159     Swal.fire({
160       title: "Création de l'événement",
161       text: "Vous devez choisir si l'événement est public ou privé",
162       icon: "error",
```

```
163       timer: 1500,
164       showConfirmButton: false
165     });
166     return;
167   }
168   //#endregion
169
170   $.ajax({
171     type: "post",
172     url: "../App/php/createEvent.php",
173     // pour l'upload de fichier
174     contentType: false,
175     processData: false,
176     data: formdata,
177     dataType: "json",
178     success: (response) => {
179       switch (response.ReturnCode) {
180         case 0:
181           Swal.fire({
182             title: "Création de l'événement",
183             text: response.Success,
184             icon: "success",
185             timer: 1300,
186             showConfirmButton: false,
187           }).then(() => {
188             EventReminder();
189             window.location.href = './index.php'
190           });
191           break;
192         case 1:
193         case 2:
194         case 3:
195         case 4:
196           Swal.fire({
197             title: "Création de l'événement",
198             text: response.Error,
199             icon: "error",
200             timer: 1500,
201             showConfirmButton: false
202           });
203           break;
204       }
205     },
206     error: (error) => {
207       console.error(error);
208     }
209   });
210 }
211
212 /**
213  * @author Hoarau Nicolas
214  * @date 27.05.20
215  * @brief Fonction qui récupère les tous les utilisateur à part le créateur de
    l'événement
216  * @return le tableau de guests
```

```
 * @version 1.0
 */
function GetGuestList() {
  if (event)
    event.preventDefault();

  let guests = [];

  // Récupère les invités
  return fetch("../App/php/getGuestList.php", {
    method: 'POST',
    headers: {
      'Accept': 'application/json',
      'Content-Type': 'application/json'
    }
  }).then(response => response.json())
    .then(async data => {
      let html = `<div class="menu"><ul>`;

      $.each(data, (key, user) => {
        html += `<li>
       <label>
         <input type="checkbox" value="${user.idUser}">
         <span class="icon"></span>
         <span class="list">${user.pseudo}</span>
       </label>
     </li>`;
      });

      html += `</ul></div>`;

      // Attend la validation de la modal par l'utilisateur et récupère les invités coché
      guests = await Swal.fire({
        title: "Liste d'invités",
        html: html,
        showCancelButton: true,
        cancelButtonText: 'Annuler',
        confirmButtonText: 'Valider'
      }).then((result) => {
        let modalGuest = [];
        // Récupère les case cochées de la modal
        let pathToLi = Swal.getContent().childNodes[0].childNodes[0].children[0].children;

        // Pour chaque <li>
        for (let i = 0; i < pathToLi.length; i++) {
          const checkbox = pathToLi[i].children[0].children[0];

          if (checkbox.checked == true) {
            modalGuest.push(checkbox.value)
          }
        }
        return modalGuest;
      });
```

```
270       return guests;
271     });
272 }
273
274 /**
275  * @author Hoarau Nicolas
276  * @date 28.05.20
277  * @brief Fonction qui récupère les événement publique qui ne sont pas encore passé
278  * @version 1.0
279  */
280 function GetPublicEvents() {
281   $.ajax({
282     type: "post",
283     url: "../App/php/getPublicEvents.php",
284     dataType: "json",
285     success: (response) => {
286       ShowPublicEvents(response);
287     },
288     error: (error) => {
289       console.error(error);
290     }
291   });
292 }
293
294 /**
295  * @author Hoarau Nicolas
296  * @date 28.05.20
297  * @brief Fonction qui récupère les événement publique qui ne sont pas encore passé,
    ainsi que les événements auxquels l'utilisateur participe et est invités
298  * @version 1.0
299  */
300 function GetManageEvents() {
301   $.ajax({
302     type: "post",
303     url: "../App/php/getManageEvents.php",
304     dataType: "json",
305     success: (response) => {
306       ShowManageEvents(response);
307     },
308     error: (error) => {
309       console.error(error);
310     }
311   });
312 }
313
314 /**
315  * @author Hoarau Nicolas
316  * @date 28.05.20
317  * @brief Fonction qui affiche les événements reçus
318  * @param {array} events
319  * @version 1.0
320  */
321 function ShowPublicEvents(data) {
322   console.log(data);
323
```

```
324   let html = `<div class="card-container">`;
325
326   for (let i = 0; i < data.events.length; i++) {
327     const event = data.events[i];
328     html += `<div class="card" id="${event.idEvenement}"
    onclick="window.location.href = './event.php?idEvent=${event.idEvenement}'">
329                 <div class="card-image"><img src="${event.urlImage}"></div>
330                 <div class="card-text">
331                 <h2>${event.nom}</h2>
332                 <span class="date">Par ${event.pseudo} <br/> de ${event.dateDebut} à
    ${event.dateFin}</span>
333                 <p>${event.descriptif}</p>
334               </div></div>`;
335
336     if (i % 5 == 0 && i != 0)
337       html += `</div><div class="card-container">`;
338   }
339
340   $("#events").html(html);
341 }
342
343 /**
344  * @author Hoarau Nicolas
345  * @date 28.05.20
346  * @brief Fonction qui récupère les événement publique qui ne sont pas encore passé,
    ainsi que les événements auxquels l'utilisateur participe et est invités
347  * @version 1.0
348  */
349 function ShowManageEvents(data) {
350   let html = ``;
351
352   // Object.keys(data).length parce data.length retourne 'undefined'
353   for (let i = 0; i < Object.keys(data).length - 1; i++) {
354     const eventZone = Object.values(data)[i];
355
356     switch (i) {
357       case 0:
358         html += `<h3>Événements auxquels vous pourrez participer</h3>
359         <div class="card-container">`;
360         break;
361       case 1:
362         html += `</div><h3>Événements auxquels vous êtes invités</h3>
363         <div class="card-container">`;
364         break;
365       case 2:
366         html += `</div><h3>Événements auxquels vous participez</h3>
367         <div class="card-container">`;
368         break;
369       case 3:
370         html += `</div><h3>Événements créé</h3>
371         <div class="card-container">`;
372         break;
373     }
374
375     if (eventZone.length === 0)
```

```
376       html += `<p style="color:#999;">Vous n'avez pas d'événements dans cette
    catégorie</p></div>`;
377
378     for (let j = 0; j < eventZone.length; j++) {
379       const event = eventZone[j];
380
381       html += `<div class="card" id="${event.idEvenement}">
382       <div class="card-image" onclick="window.location.href =
    './event.php?idEvent=${event.idEvenement}'"><img src="${event.urlImage}"></div>
383       <div class="card-text">
384         <h2>${event.nom}</h2>
385         <span class="date">Par ${event.pseudo} <br/> de ${event.dateDebut} à
    ${event.dateFin}</span>
386         <p>${event.descriptif}</p>
387       </div>`;
388
389       if (data.nickname != event.pseudo) {
390         if (i === 2) {
391           html += `<button id="paricipate" class="btn"
    onclick="Participate(${event.idEvenement}, '${event.nom}', '${event.dateDebut}',
    '${event.dateFin}')">Se désinscrire</button></div>`;
392         } else {
393           html += `<button id="paricipate" class="btn"
    onclick="Participate(${event.idEvenement}, '${event.nom}', '${event.dateDebut}',
    '${event.dateFin}')">S'inscrire</button></div>`;
394         }
395       } else {
396         html += `</div>`;
397       }
398
399       if (j % 5 == 0 && j != 0)
400         html += `</div><div class="card-container">`;
401     }
402   }
403   html += `</div>`;
404   $("#events").html(html);
405 }
406
407 /**
408  * @author Hoarau Nicolas
409  * @date 28.05.20
410  * @brief Fonction qui inscrit ou désinscris l'utilisateur
411  * @param {*} btn Les informations du bouton cliqué
412  * @version 1.0
413  */
414 function Participate(idEvent, eventName, beginDate, endDate) {
415   $.ajax({
416     type: "post",
417     url: "../App/php/willParticipate.php",
418     data: {
419       "idEvent": idEvent,
420       "eventName": eventName,
421       "beginDate": beginDate,
422       "endDate": endDate
423     },
```

```
424     dataType: "json",
425     success: (response) => {
426       switch (response.ReturnCode) {
427         case 0:
428           Toast.fire({
429             icon: 'success',
430             title: response.Success
431           }).then(() => {
432             if (pageName == "index.php" || pageName == "") {
433               GetPublicEvents()
434             } else if (pageName == "manageEvents.php") {
435               GetManageEvents()
436             } else {
437               GetEventData(idEvent);
438             }
439           });
440           break;
441         case 1:
442           Toast.fire({
443             icon: 'error',
444             title: response.Error
445           });
446           break;
447       }
448     },
449     error: (error) => {
450       console.error(error);
451     }
452   });
453 }
454
455 /**
456  * @author Hoarau Nicolas
457  * @date 02.06.20
458  * @brief Fonction qui récupère les infrormations de l'id d'événement reçus
459  * @param {int} idEvent
460  */
461 function GetEventData(idEvent) {
462   $.ajax({
463     type: "post",
464     url: "../App/php/getEventData.php",
465     data: { idEvent: idEvent },
466     dataType: "json",
467     success: (response) => {
468       switch (response.ReturnCode) {
469         case 0:
470           ShowEventData(response.Data);
471           break;
472         case 1:
473           window.location.href = './index.php'
474           break;
475         case 404:
476           window.location.href = './404.php'
477           break;
478       }
```

```
    },
    error: (error) => {
      console.error(error);
    }
  });
}

/**
 * @author Hoarau Nicolas
 * @date 02.06.20
 * @brief Fonction qui affiche les données d'un événement choisis
 * @param {array} data
 */
function ShowEventData(data) {
  document.title = data.nom;

  let html = `<div class="event-data">
  <img src="${data.urlImage}">
  <h2 class="title">${data.nom}</h2>
  <div class="div place">${data.prive == 1 ? 'Privé' : 'Publique'}</div>
  <div class="div date">De ${data.dateDebut} à ${data.dateFin}<br> Par ${data.pseudo}
</div>
  <div class="div">${data.descriptif}</div>
  <div class="div place">${data.lieu}</div>
  <div class="div">${data.nbGuest} sur ${data.nbMaxParticipant} participants <a
id="showGuests" onclick="GetEventGuestList(${data.idEvenement})">Voir les
participants</a></div>`;

  if (data.isLogged === true) {
    if (data.nickname === data.pseudo) {
      if (data.prive == 1)
        html += `<div class="div"><a id="showInvited"
onclick="GetInvited(${data.idEvenement})"> Voir les invités</a> <a id="showInvited"
onclick="GetEditGuestList(${data.idEvenement})">Modifier la liste des invités</a>
</div>`;

      html += `<button class="btn btn-edit" onclick="window.location.href =
'./editEvent.php?idEvent=${data.idEvenement}'">Modifier</button>
      <button class="btn btn-delete" onclick="DeleteEvent(${data.idEvenement},
'${data.nom}', '${data.prive}')">Supprimer</button>`;
    } else {
      if (data.participating == 1) {
        html += `<button class="btn" onclick="Participate(${data.idEvenement},
'${data.nom}', '${data.dateDebut}', '${data.dateFin}')">Se désinscrire</button>`;
      } else {
        if (data.nbGuest < data.nbMaxParticipant) {
          html += `<button class="btn" onclick="Participate(${data.idEvenement},
'${data.nom}', '${data.dateDebut}', '${data.dateFin}')">S'inscrire</button>`;
        }
      }
    }
  } else {
    Toast.fire({
      icon: 'info',
      title: "Vous ne pouvez pas participer à l'événement sans être connecté"
```

```
524     });
525   }
526
527   html += `</div>`;
528
529   $("#eventData").html(html);
530 }
531
532 /**
533  * @author Hoarau Nicolas
534  * @date 02.06.20
535  * @brief Fonction qui récupère les les participants de l'événement voulus
536  * @param {int} idEvent
537  */
538 function GetEventGuestList(idEvent) {
539   $.ajax({
540     type: "post",
541     url: "../App/php/getEventGuestList.php",
542     data: { idEvent: idEvent },
543     dataType: "json",
544     success: (response) => {
545       ShowEventGuestList(response.Data);
546     },
547     error: (error) => {
548       console.error(error);
549     }
550   });
551 };
552
553 /**
554  * @author Hoarau Nicolas
555  * @date 02.06.20
556  * @brief Fonction qui affiche participants d'un événement
557  * @param {array} guestList
558  */
559 function ShowEventGuestList(guestlist) {
560   let html = `<div class="menu">
561   <ul>`;
562
563   for (let i = 0; i < guestlist.length; i++) {
564     const guest = guestlist[i];
565     html += `<li>
566     <label>
567       <span class="list">${guest.pseudo}</span>
568     </label>
569   </li>`;
570   }
571
572   html += `</ul></div>`;
573
574   Swal.fire({
575     title: "Liste des participants",
576     html: html,
577     confirmButtonText: 'Valider'
578   });
```

```
579 }
580
581 /**
582  * @author Hoarau Nicolas
583  * @date 03.06.20
584  * @brief Fonction qui envoie un mail de rappel pour les événements
585  */
586 function EventReminder() {
587   $.ajax({
588     type: "post",
589     url: "../App/php/eventReminder.php",
590     dataType: "json",
591     success: (response) => {
592       console.log('email correctly send');
593     },
594     error: (error) => {
595       console.error(error);
596     }
597   });
598 }
599
600 /**
601  * @author Hoarau Nicolas
602  * @date 03.06.20
603  * @brief Fonction qui envoie les données pour supprimer un événement
604  * @param {int} idEvent
605  */
606 function DeleteEvent(idEvent, eventName, private) {
607   console.log(private);
608
609   $.ajax({
610     type: "post",
611     url: "../App/php/isEventDeletable.php",
612     data: {
613       idEvent: idEvent,
614       eventName: eventName
615     },
616     dataType: "json",
617     success: (response) => {
618       switch (response.ReturnCode) {
619         case 0:
620           Swal.fire({
621             title: "Suppression de l'événement",
622             text: response.Success,
623             icon: "question",
624             showConfirmButton: true,
625             confirmButtonText: 'Valider',
626             showCancelButton: true,
627             cancelButtonText: 'Annuler',
628             reverseButtons: true
629           }).then((result) => {
630             if (result.value) {
631               $.ajax({
632                 type: "post",
633                 url: "../App/php/deleteEvent.php",
```

```
634                 data: {
635                   idEvent: idEvent,
636                   eventName: eventName,
637                   private: private
638                 },
639                 dataType: "json",
640                 success: (data) => {
641                   switch (data.ReturnCode) {
642                     case 0:
643                       Swal.fire({
644                         title: "Suppression de l'événement",
645                         text: response.Success,
646                         icon: "success",
647                         timer: 1300,
648                         showConfirmButton: false,
649                       }).then(() => {
650                         window.location.href = './index.php'
651                       });
652                       break;
653                     case 1:
654                       Swal.fire({
655                         title: "Suppression de l'événement",
656                         text: response.Error,
657                         icon: "error",
658                         timer: 1500,
659                         showConfirmButton: false,
660                       });
661                       break;
662                   }
663                 },
664                 error: (error) => {
665                   console.error(error);
666                 }
667               });
668             }
669           });
670           break;
671         case 1:
672           Swal.fire({
673             title: "Suppression de l'événement",
674             text: response.Error,
675             icon: "error",
676             timer: 1500,
677             showConfirmButton: false
678           });
679           break;
680       }
681     },
682     error: (error) => {
683       console.error(error);
684     }
685   });
686 }
687
688 /**
```

```
 * @author Hoarau Nicolas
 * @date 04.06.20
 * @brief Fonction qui envoie les données pour modifier un événement
 * @param {*} event
 * @param {int} idEvent
 */
function EditEvent(event, idEvent) {
  if (event)
    event.preventDefault();

  let formdata = new FormData();
  let eventName = $("#eventName").val();
  let place = $("#place").val();
  let eventDescription = $("#eventDescription").val();
  let nbMaxGuest = $("#nbMaxGuest").val();
  let img = document.getElementById("img").files[0];
  let beginningDate = new Date($("#beginningDate").val());
  let endDate = new Date($("#endDate").val());
  let beginningTime = $("#beginningTime").val();
  let endTime = $("#endTime").val();
  let now = new Date();

  beginningDate.setHours(beginningTime.split(':')[0], beginningTime.split(':')[1], 0,
0);
  endDate.setHours(endTime.split(':')[0], endTime.split(':')[1], 0, 0);
  now.setHours(now.getHours(), now.getMinutes(), 0, 0);
  formdata.append('idEvent', idEvent);
  //#endregion

  //#region Conditions
  if (eventName.length == 0) {
    $("#eventName").focus();
    return;
  } else {
    formdata.append("eventName", eventName);
  }

  if (eventDescription.length == 0) {
    $("#eventDescription").focus();
    return;
  } else {
    formdata.append("eventDescription", eventDescription);
  }

  if (place.length == 0) {
    $("#place").focus();
    return;
  } else {
    formdata.append("place", place);
  }

  if (nbMaxGuest == 0 || nbMaxGuest == "") {
    $("#nbMaxGuest").focus();
    return;
  } else {
```

```
743     formdata.append("nbMaxGuest", nbMaxGuest);
744   }
745
746   if (beginningDate === null) {
747     $("#beginningDate").focus();
748     return;
749   } else if (beginningDate < now) {
750     Swal.fire({
751       title: "Création de l'événement",
752       text: "La date choisis est déjà passé",
753       icon: "error",
754       timer: 1500,
755       showConfirmButton: false
756     });
757     return;
758   } else {
759
760     formdata.append("beginningDate", beginningDate.toLocaleDateString());
761     formdata.append("beginningTime", beginningTime)
762   }
763
764   if (endDate === null) {
765     $("#endDate").focus();
766     return;
767   } else if (endDate < beginningDate) {
768     Swal.fire({
769       title: "Création de l'événement",
770       text: "La date choisis est avant le début de l'événement",
771       icon: "error",
772       timer: 1500,
773       showConfirmButton: false
774     });
775     return;
776   } else {
777     formdata.append("endDate", endDate.toLocaleDateString());
778     formdata.append("endTime", endTime);
779   }
780
781   if (typeof img !== 'undefined') {
782     formdata.append("img", img);
783   }
784   //#endregion
785
786   $.ajax({
787     type: "post",
788     url: "../App/php/editEvent.php",
789     // pour l'upload de fichier
790     contentType: false,
791     processData: false,
792     data: formdata,
793     dataType: "json",
794     success: (response) => {
795       switch (response.ReturnCode) {
796         case 0:
797           Swal.fire({
```

```
798             title: "Création de l'événement",
799             text: response.Success,
800             icon: "success",
801             timer: 1300,
802             showConfirmButton: false,
803           }).then(() => {
804             window.location.href = './index.php'
805           });
806           break;
807         case 1:
808         case 2:
809         case 3:
810         case 4:
811           Swal.fire({
812             title: "Création de l'événement",
813             text: response.Error,
814             icon: "error",
815             timer: 1500,
816             showConfirmButton: false
817           });
818           break;
819       }
820     },
821     error: (error) => {
822       console.error(error);
823     }
824   });
825 }
826
827 /**
828  * @author Hoarau Nicolas
829  * @date 05.06.20
830  * @brief Fonction qui récupère les invités à un événement
831  * @param {int} idEvent
832  */
833 function GetInvited(idEvent) {
834   $.ajax({
835     type: "post",
836     url: "../App/php/getInvited.php",
837     data: { idEvent: idEvent },
838     dataType: "json",
839     success: (response) => {
840       switch (response.ReturnCode) {
841         case 0:
842           ShowEventInvitedList(response.Data);
843           break;
844         case 1:
845           Swal.fire({
846             title: "Liste des invités",
847             text: response.Error,
848             icon: "error",
849             timer: 1500,
850             showConfirmButton: false
851           });
852           break;
```

```
853       }
854     },
855     error: (error) => {
856       console.error(error);
857     }
858   });
859 }
860
861 /**
862  * @author Hoarau Nicolas
863  * @date 05.06.20
864  * @brief Fonction qui affiche les invités à événement
865  * @param {array} invitedList
866  */
867 function ShowEventInvitedList(invitedList) {
868   let html = `<div class="menu"><ul>`;
869
870   for (let i = 0; i < invitedList.length; i++) {
871     const invited = invitedList[i];
872     html += `<li>
873     <label>
874       <span class="list">${invited.pseudo}</span>
875     </label>
876   </li>`;
877   }
878
879   html += `</ul></div>`;
880
881   Swal.fire({
882     title: "Liste des invités",
883     html: html,
884     confirmButtonText: 'Valider'
885   });
886 }
887
888 /**
889  * @author Hoarau Nicolas
890  * @date 05.06.20
891  * @brief Fonction qui récupère les utilisateurs en disant s'il participe ou non
    l'événement donnée
892  * @param {int} idEvent
893  */
894 function GetEditGuestList(idEvent) {
895   $.ajax({
896     type: "post",
897     url: "../App/php/getEditGuestList.php",
898     data: { idEvent: idEvent },
899     dataType: "json",
900     success: (response) => {
901       switch (response.ReturnCode) {
902         case 0:
903           ShowEditGuestList(response.Data, idEvent);
904           break;
905         case 1:
906           Swal.fire({
```

```
907             title: "Liste des invités",
908             text: response.Error,
909             icon: "error",
910             timer: 1500,
911             showConfirmButton: false
912           });
913           break;
914       }
915     },
916     error: (error) => {
917       console.error(error);
918     }
919   });
920 }
921
922 /**
923  * @author Hoarau Nicolas
924  * @date 05.06.20
925  * @brief Fonction qui affche la modal pour la modification de la liste des invités e
    envoie les données de modification
926  * @param {array} data
927  * @param {int} idEvent
928  */
929 function ShowEditGuestList(data, idEvent) {
930   let html = `<div class="menu"><ul>`;
931
932   $.each(data, (key, user) => {
933     html += `<li>
934    <label>
935      <input type="checkbox" value="${user.idUser}" ${user.invited == 1 ? 'checked' :
    ''}>
936      <span class="icon"></span>
937      <span class="list">${user.pseudo}</span>
938    </label>
939  </li>`;
940   });
941
942   html += `</ul></div>`;
943
944   Swal.fire({
945     title: "Liste d'invités",
946     html: html,
947     showCancelButton: true,
948     cancelButtonText: 'Annuler',
949     confirmButtonText: 'Valider'
950   }).then((result) => {
951     if (result.value) {
952       let editGuestlist = [];
953       // Récupère les case cochées de la modal
954       let pathToLi =
    Swal.getContent().childNodes[0].childNodes[0].children[0].children;
955
956       // Pour chaque <li>
957       for (let i = 0; i < pathToLi.length; i++) {
958         const checkbox = pathToLi[i].children[0].children[0];
```

```
959
960         // regarde els différences avec la liste de base
961         if (data[i].invited != checkbox.checked) {
962           editGuestlist.push([checkbox.value, checkbox.checked])
963         }
964       }
965
966       if (editGuestlist.length > 0) {
967         $.ajax({
968           type: "post",
969           url: "../App/php/editGuestList.php",
970           data: {
971             editGuetsList: editGuestlist,
972             idEvent: idEvent
973           },
974           dataType: "json",
975           success: (response) => {
976             switch (response.ReturnCode) {
977               case 0:
978                 Swal.fire({
979                   title: "Liste d'invités",
980                   text: response.Success,
981                   icon: "success",
982                   timer: 1300,
983                   showConfirmButton: false
984                 });
985                 break;
986               case 1:
987                 Swal.fire({
988                   title: "Liste d'invités",
989                   text: response.Error,
990                   icon: "error",
991                   timer: 1300,
992                   showConfirmButton: false
993                 });
994                 break;
995             }
996           },
997           error: (error) => {
998             console.error(error);
999           }
1000        });
1001      }
1002    }
1003  });
```