

```

512
513 /**
514  * This function replaces the set of divisions by the stated divisions. The function guarantees that the inverse membership is handled correctly.
515  * @param divisionRepository The division repository needed to save the altered divisions.
516  * @param eventRepository The event repository needed to save the altered events.
517  * @param divs The new list of divisions for the user.
518  */
519 public void replaceDivisions(DivisionRepository divisionRepository, EventRepository eventRepository, List<Division> divs)
520 {
521     logger.debug("[{}] Replacing division set", this);
522
523     List<Division> finalDivisions = DivisionHelper.getExpandedSetOfDivisions(divs, divisionRepository);
524     List<Division> oldDivisions = divisions;
525
526     if((finalDivisions == null || finalDivisions.isEmpty()) && (oldDivisions == null || oldDivisions.isEmpty()))
527     {
528         logger.debug("[{}] Division sets before and after are both empty", this);
529         divisions = new ArrayList<>();
530     } else if(finalDivisions == null || finalDivisions.isEmpty())
531     {
532         logger.debug("[{}] Division set after is empty, before is not. Removing membership subscription from old divisions", this);
533         oldDivisions.stream().forEach(div -> div.removeMember(this));
534         divisionRepository.save(oldDivisions);
535
536         //Updating events, affected by division change
537         oldDivisions.parallelStream().forEach(div -> {
538             List<Event> changedEvents = eventRepository.findByInvitedDivision(div);
539             changedEvents.parallelStream().forEach(event -> event.updateInvitedUser(divisionRepository));
540             eventRepository.save(changedEvents);
541         });
542         divisions = new ArrayList<>();
543     } else if(oldDivisions == null || oldDivisions.isEmpty())
544     {
545         logger.debug("[{}] Division set before is empty, after is not. Adding membership subscription to new divisions", this);
546         finalDivisions.stream().forEach(div -> div.addMember(this));
547         divisionRepository.save(finalDivisions);
548
549         //Updating events, affected by division change
550         finalDivisions.parallelStream().forEach(div -> {
551             List<Event> changedEvents = eventRepository.findByInvitedDivision(div);
552             changedEvents.parallelStream().forEach(event -> event.updateInvitedUser(divisionRepository));
553             eventRepository.save(changedEvents);
554         });
555         divisions = finalDivisions;
556     } else
557     {
558         logger.debug("[{}] Division set after and before are not empty. Applying changed membership subscriptions", this);
559         List<Division> intersect = finalDivisions.stream().filter(oldDivisions::contains).collect(Collectors.toList()); //These items are already in the list, and do not need to be modified
560
561         //Collecting changed division for batch save
562         List<Division> changedDivisions = Collections.synchronizedList(new ArrayList<>());
563
564         //Removing membership from removed divisions
565         oldDivisions.parallelStream()
566             .filter(div -> !intersect.contains(div))
567             .forEach(div -> {
568                 div.removeMember(this);
569                 changedDivisions.add(div);
570             });
571
572         //Adding membership to added divisions
573         finalDivisions.parallelStream()
574             .filter(div -> !intersect.contains(div))
575             .forEach(div -> {

```

```
576         div.addMember(this);
577         changedDivisions.add(div);
578     });
579
580     divisionRepository.save(changedDivisions);
581
582     //Updating events, affected by division change
583     changedDivisions.parallelStream().distinct().forEach(div -> {
584         List<Event> changedEvents = eventRepository.findByInvitedDivision(div);
585         changedEvents.parallelStream().distinct().forEach(event -> event.updateInvitedUser(divisionRepository));
586         eventRepository.save(changedEvents);
587     });
588     divisions = finalDivisions;
589 }
590 }
```