SOLUTION

API 1 : Récupère tous les événements de la base. api/v1/events?isLive=true

La solution est la suivante : Si on reçoit le paramètre isLive=true, alors on fait appel à une méthode sql écrite par nous. Dans le cas contraire on utilise findAll de jpa. On passe le résultat à un mappeur qui retourne le résultat.

```
List<Event> events = null ;
if(isLive) {
  events = eventRepository.findByMarkets_Selections_State(SelectionState.OPENED);
}else {
  events = eventRepository.findAll();
}
return EventMapper.mapEventsToEventsDto(events);
```

API 2 : Récupère toutes les sélections d'un événément: api/v1/events/{id}/selections?{state}=state

Pareil que la première API :

```
Optional<Event> event = eventRepository.findById(eventId);
if(event.isEmpty()) {
 throw new CustomException("Evénement introuvable",  ExceptionType.EVENTS_NOT_FOUND);
}

List<Selection> selections = null;

if(state == null) {
  selections = new ArrayList<Selection>();
  for(Market market: event.get().getMarkets()) {
    if(market.getSelections() != null) {
       selections.addAll(market.getSelections());
    }
  }
}else {
 selections = eventRepository.findByIdAndMarkets_Selections_State(eventId, state);
}

return SelectionMapper.mapSelectionsToSelectionsDto(selections);
```

API 3 : Enregistre un pari. api/v1/bets/add

http://localhost:8887/api/v1/bets/add

POST,

BODY : {

"selectionId" : "5",

"cote" : 4.4,

"mise" : 1

}

D'après la spécification un utilisateur ne peut pas faire deux paris simultanément : il faut s'assurer que la synchronisation soit ok.

1ere solution : Méthode d'authentification : ne pas permettre qu'un utilisateur se connecte, depuis plusieurs appareils simultanément (utilisé un champ token dans la base de donnée pour avoir un seul token valide). Mais un utilisateur peut utiliser plusieurs instances de Postman avec le même token.

2iem solution : Utilisé une méthode synchronized pour exécuter la section critique : mais ceci pourra créer un temps d'attente élevé coté utilisateur, car ceci bloquera tous les utilisateurs.

3ieme solution et celle utilisée : Synchronisation niveau data base. Spring boot met a disposition plusieurs mécanisme pour la synchronisation. Nous avons utilisé `LockModeType.PESSIMISTIC_READ` dans le repository pour mettre un lock sur la ligne en cours d'utilisation.

POUR LA GESTION DES EXCEPTIONS :

POURQUOI REDEFINIR LES CONSTANTES QUI EXISTENT DEJA ?
`CUSTOMER_NOT_FOUND(HttpStatus.NOT_FOUND), Pour les évents ou sélections not found il faut redéfinir deux autres constantes ?`

La constante `HttpStatus` n'a pas les valeur 600, 601, 602 pour pouvoir retourner ces codes d'erreurs nous avons modifié manuellement l'objet httpResponse