Database schema for events

💡 In conclusion:

The **events** table should be a realtime table. Every single change in any of its attributes should notify all users linked to it via a linking table.

Also for other features of the app (ie. canceling an event) this is necessary.

We assume that all users are already logged in.

1. *User1*, *User2* both register to *Event1*
   - How does it happen?

     User clicks on an invite link, received from an organizer and lands on the event page.

     User then clicks on the "register" button and picks either *GroupA* or *GroupB*.
   - What does this require?

     Event start-date must be in the future.

     The **registered** table must not contain [*User1*, *Event1*, <bool>] or [*User2*, *Event2*, <bool>].
   - What effect does this have?

     User gets added to the **registered** table

       schema: [user, event, inEvent]

       creates [*User1*, *Event1*, false]

       creates [*User2*, *Event1*, false]

     Event will appear on users dashboard page (once an event starts, it also gets an “I’m there” button there).

2. *User1*, *User2* both participate in *Event1*
   - How does it happen?

     User clicks on the “I’m there” button for *Event1* in the dashboard page.
   - What does this require?

     The attribute *ended* of the *Event1* entity in the **events** table must be “false”.

       This means that the organizer hasn’t yet ended the event (this can happen any time).

     Event1’s start-dateTime must be in the present or the past.

     The **registered** table must contain [*User1*, *Event1*, false] or [*User2*, *Event2*, false].
   - What effect does this have?

     In the **registered** table:

       schema: [user, event, inEvent]

       [*User1*, *Event1*, false] updated to [*User1*, *Event1*, true]

       [*User2*, *Event1*, false] updated to [*User2*, *Event1*, true]

     The button “leave event” then appears in the navigation menu, that the user can use to leave the event and set the *participating* value to “false” again (but the user can join back any time).

Users who are participating, all get forwarded to the page that allows them to enter another persons ID or scan their QR-Code (which contains their ID but encoded).

This can be done after giving them an optional introduction to how speed dating works.

3. *User1, User2* create the pairing *Pairing{1,2}*
   - How does it happen?

     Any of the users scans the other users QR-code or enters their ID and the other user clicks “yes” when the “accept pairing with user1283u10?” message occurs.
   - What does this require?

     User1, User2 must be in different groups.

     The pairing [*User1*, *User2*, *Event1*, <bool>] must not be in the **pairings** table.

     The attribute *ended* of the *Event1* entity in the **events** table must be “false”.

     Event1’s start-dateTime must be in the present or the past.

     The attribute *timeDateRoundEnd* of the *Event1* entity in the **events** table must be “null”.

     This means that there currently is no running round.
   - What effect does this have?

     Pairing{1,2} entity gets added to **pairings** table.

     schema: [user1, user2, event, match]

     creates [*User1*, *User2*, *Event1*, false]

4. *Event1Organizer* starts round, pairings get locked and the timer starts
   - How does it happen?

     Event1Organizer selects a timespan for the next round and presses the “start round” button.
   - What does this require?

     The attribute *ended* of the *Event1* entity in the **events** table must be “false”.

     Event1’s start-dateTime must be in the present or the past.

     The attribute *timeDateRoundEnd* of the *Event1* entity in the **events** table must be “null”.

     This means that there currently is no running round.

     The attribute *timeDateLastRoundEnd* of the *Event1* entity in the **events** table must be at least 2 minutes ago. This means that there currently are no users still deciding whether they like the other person or not.
   - What effect does this have?

     The attribute *timeDateRoundEnd* of the *Event1* entity in the **events** table gets updated from “null” to a new value, based on current time and selected timespan.

     All users get notified and their timer starts.

5. *Event1Organizer* ends round manually (or the browser send the request automatically because the time ended), pairings get unlocked and the timer stops
   - How does it happen?

     Event1Organizer presses the “start round” button.
   - What does this require?

     The attribute *ended* of the *Event1* entity in the **events** table must be “false”.

     Event1’s start-dateTime must be in the present or the past.

The attribute *timeDateRoundEnd* of the *Event1* entity in the **events** table must not be “null”.

This means that there currently is a running round.

- What effect does this have?

  When the round ends, then the attribute *timeDateRoundEnd* of the *Event1* entity in the **events** table gets updated back to “null”.

  All users get notified again.

  The attribute *timeDateLastRoundEnd* of the *Event1* entity in the **events** table gets updated to the current time.

  Users are then have 2 minutes to decide, whether they liked the other person or not. If they are too slow, the default response is “no”.

  If both answer with “yes”, then the *Pairing{1,2}* entity in the **pairings** table gets updated:

  schema: [user1, user2, event, match]

  [*User1*, *User2*, *Event1*, false] updated to [*User1*, *User2*, *Event1*, true]

6. *Event1Organizer* does not end event

   Then we continue from step 5.

7. *Event1Organizer* ends event with a preset time, after which participants should see their matches

   - How does it happen?

     Event1Organizer presses the “end event” button.

     Then they must select a timespan after which all participants receive mails with their matches.

   - What does this require?

     The attribute *ended* of the *Event1* entity in the **events** table must be “false”.

     Event1’s start-dateTime must be in the present or the past.

     The attribute *timeDateRoundEnd* of the *Event1* entity in the **events** table must be “null”.

     This means that there currently is no running round.

   - What effect does this have?

     The attribute *ended* of the *Event1* entity in the **events** table gets updated to “true”.

     All users get notified and they get forwarded to a goodbye page.

     In the **registered** table gets updated:

     schema: [user, event, inEvent]

     [*User1*, *Event1*, true] updated to [*User1*, *Event1*, false]

     [*User2*, *Event1*, true] updated to [*User2*, *Event1*, false]

     Then after the given timespan all matches [*user1*, *user2*, *event*, true] in the **pairings** table for each person, receive their partners contact details.