**Project Summary**

What is your name?

Carlos Sanchez

What E-mail address do you use to sign in to Udacity?

cafesanu@gmail.com

What is your App ID? Please also provide a link to your deployed app for our convenience.

App ID: cafesanu-gae

App URL: https://cafesanu-gae.appspot.com

Is there any other important information that you would want your project evaluator to know?

No, other that the details in the next section

**Design choices**

Sessions are very similar to conferences, I did not change much other that actually storing the conference key as the parent of each session, the reason is that I read an example on the GAE documentation where it was done this way, as opposite of storing the id (like the way it was done during the video lectures). I defined speaker as a simple String, I could not find a reason of creating a kind for a speaker given that it would only have one attribute (name), and it wouldn't make sense having another entity as its ancestor or child.

The indexes I created are: name, speaker, and type, and time given that those are the ones field I added in the queries, but more could be added if you'd like a more powerful getConferenceSessionsQueryForm (described above)

**Extra stuff I added not required by the rubric :**

- API method: getConferenceSessionsQueryForm. I added a SessionQueryForm similar to ConferenceQueryForm that can query almost anything that is indexed (name, speaker, and type, and time for now, but I could add more indexes).
- When creating a session, I added sending an email with the session info as well.
- API method: deleteSessionFromWhishlist: Deleted a session from wishlist.

**Additional Queries**

1. API method: getSessionsByDate. Searches by date. When creating a session via API, date needs to be input like "2014-07-21T00:00:00.000Z", same for using this query: please enter date the same way you entered it when creating session.
   Code(Included in API code):

```
/**
 * Returns a list of sessions with the specified date. In order to receive
 * the websafeConferenceKey via the JSON params, uses a POST method.
 *
 * @param date
 *            The date session starts
 * @return a list of Session with the specified date
 */
@ApiMethod(
    name = "getSessionsByDate",
    path = "getSessionsByDate",
    httpMethod = HttpMethod.POST
)
public List<Session> getSessionsByDate(@Named("date") final Date date) {
    Query<Session> q = ofy().load().type(Session.class)
                                    .filter("date =", date);
    return q.list();
}
```

2. API method: getSessionsByTimeRange. If user want's to see if there are session within a time frame
   Code(Included in API code):

```
    /**
     * Returns a list of sessions within the specified time. In order to
     * receive the websafeConferenceKey via the JSON params, uses a POST method.
     *
     * @param after
     *            The minimum start time
     * @param before
     *            The maximum start time
     *
     * @throws IllegalArgumentException
     *            If times are not in HH:MM format
     *
     * @return a list of Session with the specified time
     */
    @ApiMethod(
        name = "getSessionsByTimeRange",
        path = "getSessionsByTimeRange",
        httpMethod = HttpMethod.POST
    )
    public List<Session> getSessionsByTimeRange(@Named("after") final String after, @Named("before") final String
before)
                    throws IllegalArgumentException
    {

        boolean validAfter = Time24HoursValidator.validate(after);
        boolean validBefore = Time24HoursValidator.validate(before);
        if(!validAfter || !validBefore ){
            throw new IllegalArgumentException("Time error. Enter times in format HH:MM.");
        }
        Query<Session> q = ofy().load().type(Session.class)
                                        .filter("time >=", after)
                                        .filter("time <=", before);
        return q.list();
    }
```

**Query related problem**
The problem of this query is that is has two inequalities for two different indexes("!=" for sessionType, and "<" for time). I actually implemented this query and can be found in the api as getSessionsBeforeTimeOtherThanType. This query queries by time less that and specific time (in format hh:mm), and then traverses the result looking for sessions which type is different that the type passed, when it finds a session with this requisite, it adds it to a result list that will be returned at the end.
Code(Included in API code):

```
    /**
     * Returns a list of sessions which type is different than type whit time
     * less than time
     *
     * @param type
     *            The session type
     * @param time
     *            The time before
     * @return a list of sessions which type is different than type whit time
     * less than time
     */
    @ApiMethod(
        name = "getSessionsBeforeTimeOtherThanType",
        path = "getSessionsBeforeTimeOtherThanType",
        httpMethod = HttpMethod.POST
    )
    public List<Session> getSessionsBeforeTimeOtherThanType(@Named("type") final String type, @Named("time") final String
time)
    {
        Query<Session> q = ofy().load().type(Session.class)
                                .filter("time <", time);
        List<Session> sessionsBeforeTime = q.list();
        List<Session> sessionsBeforeTimeOtherThanType = new ArrayList<>(0);
        for (Session s : sessionsBeforeTime) {
            if (!s.getType().equals(type)) {
                sessionsBeforeTimeOtherThanType.add(s);
            }
        }
        return sessionsBeforeTimeOtherThanType;
    }
```