User — **sessAuthPolicy** — **Meeting** — **User** — **Room**

create
{title: string}
authUser
[!loggedIn]
return Forbidden
create
{title: string}
meetingObject
return
ownerAdd
{meeting: id, user: authUser}
userObject || {}
return
find
{room: id}
roomObject
return
return {success: true}

findAll
authUser
[!loggedIn]
return Forbidden
find
{owner: authUser}
find
{room: id}
roomObject
return
meetingArr
return
return meetingArr

findOne
{meeting:id}
authUser
[!loggedIn]
return Forbidden
find
{owner: authUser,
meeting: id}
populate
{meeting: id}
userArr || []
return
[!authUser in ownerArr]
return {owner:false}
return Forbidden
find
{room: id}
roomObject
return
meetingArr
return
return meetingObj

Person

name: { type: "string"},
surname: { type: "string"},
email: { type: "string", required: true},
meetings: {collection: "Rsvp", via: "person"}

Holds User and Invitee data

Meetings → Meetings Invited To

User

person: {model: "Person", unique: true},
password: {type: "string", required: true},
meetings: {collection: "Meeting", via: "owners"},
rooms: {collection: "Room", via: "owners"},
toJSON: function() {
  var obj = this.toObject();
  delete obj.password;
  return obj;
}

A User can be authenticated on the web interface

Person → Person object
Meetings → Meetings Owned
Rooms → Meeting Rooms Owned
Password → Password used to Authenticate

Room

title: {type: "string", required: true},
location: {type: "string", required: true},
meetings: {collection: "Meeting", via: "room"},
owners: {collection: "User", via: "rooms"},
apiKey: {type: "string", required: true},
apiSecret: {type: "string", defaultsTo: generateSecret()}

This is the device that is placed outside the meeting room

Title → Meeting Room name
Location → Address of the meeting room
Meetings → All meetings taking place in the room
Owners → Users with access to create meetings in the room
apiKey → Used to identify the room when server request
apiSecret → Used to encrypt packet sent to server

Device

imei: {type: "string"},
deviceObj: {type: "json"},
rsvp: {collection: "Rsvp", via: "device"}

A device is registered first run of the app

imei → Device unique identifier
deviceObj → Summary of the type of device
Rsvp → All meetings where the device was used to RSVP Yes

Rsvp

status: {
  type: 'string',
  enum: ['pending', 'yes', 'no', 'in', 'out', 'removed'],
  defaultsTo: 'pending'
},
meeting: { model: "Meeting", via: "rsvp", required: true},
password: { type: "string", defaultsTo: generatePassword()},
person: { model: "Person", via: "meetings", required: true},
device: { model: "Device", via: "rsvp"},
logs: {collection: "Log", via: "rsvp"}

Created when person is invited

Status →
- pending: Invite Sent (->[yes || no || removed])
- yes: Invite accepted (->[in || removed])
- no: Invite declined (->[removed])
- in: Person currently in meeting (->[out || removed])
- out: Person currently out of meeting (->[in || removed])
- removed: Person was deleted from the invite list or new
            Rsvp was created for that person
Password → Used to RSVP Yes
Person → Person invited
Device → Contains the IMEI of device when they RSVP yes
Logs → Save copy of every Rsvp object update

Meeting

title: { type: "string"},
description: { type: "string"},
room: {model: "Room", via: "meetings"},
dateStart: {type: "datetime"},
dateEnd: {type: "datetime"},
owners: { collection: "User", via: "meetings"},
rsvp: { collection: "Rsvp", via: "meeting"},
logs: {collection: "Log", via: "meeting"}

The meeting object is created for every new meeting

Title → Meeting title
Description → Meeting Description
Room → Room where the meeting takes place
dateStart,dateEnd → DateTime of meeting (Begin, End)
Rsvp → An array of rsvp objects for that meeting

Logs

meeting: {model: "Meeting", via: "logs"},
rsvp: {model: "Rsvp", via: "logs"},
state: {type: "json", required: true}

Contains all the logs of RSVP changes

Meeting → Meeting in RSVP
Rsvp → Rsvp Changed
State → Current state of object

Mail

person: {model: "Person"},
emailObj: {type: "json"}

Email is sent on create of this object

Person → Person the mail is sent to
emailObj → The email obj passed to the email sender
also saves all attachments added in the emailObj