UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Faculty of Engineering.

Subject: Advanced Programming

Teacher: Carlos Andres Sierra

Project: Virtual Experience (chati)

Sergio Nicolas Mendivelso	**20231020227**
Daniel Santiago Pérez	**20231020203**

Bogotá, D.C.

10 April 2024.

Project introduction:

Virtual Xperience is a virtual event platform that helps the organization, management and the participation of these events. The users can register themselves so they can access the events they want. Also, they can create events with their own activities and assignments.

The application has a calendar, so the user can watch their schedule and programmed events and activities.

Business model:

The business model of the virtual event platforms it's based on creating new events where the organizer can add activities and assignments online. The users also can design these events.

Some functions of the platforms are the following points:

- **Create events:** The organizer can design virtual events and activities, with date, hour, description, and type.
- **Inscription management:** The platform facilitates the inscription process for the participants, some product sales and email confirmations.
- **Real time interaction:** The platform provides tools to interact with other participants in real times, like chats, forums and question and ask sessions.
- **Access to the content:** The participants can access the event material, like slides, videos, chats, and relationed themes.

To design the business model, we have to create an architecture that can manage a lot of events at time. Another important thing is the security of this project.

Business rules:

- The users (Organizers or participants) must be registered with an unique name, email and a password, so they can log in and access their information.
- The Organizers can create events and manage them, adding activities, and participants to the event.
- The participants can interact with each other, with the chats.
- The participants can interact with the material uploaded by the organizer of the event
- The users can watch their next activities in the calendar.

StakeHolders:

- Event organizers: like business and people that want to do events.
- Participants: People that want to participate in those events.

- Sponsors: Companies that can finance some great events.
- Developers: The people that can program this platform.

Tools:

Programming languages : Html, CSS,JS for frontend and Python for the backend
FrameWorks: Django, canvas.

User stories:

- □ As user (Organizer or participant), I want to register myself with a unique name and password, so I can save information and protect my privacy.
- □ As user (Organizer or participant), I want to login with my account, so I can access my information and can know in which events I am organizer or participant.
- □ As organizer, I want to create virtual events, So I can plan the events in a successful way.
- □ As organizer I want to manage all my virtual events, also add the activities and assignments, so the participants can interact with these activities.
- □ As organizer I want to share documents and videos as material to my participants.
- □ As Organizer I want to restrict the access to some users, so I can decide who can assist in my events protecting my privacy.
- □ As user I want to see my DashBoard, so I can watch my future activities.
- □ As Participant I want to Sign up to an event.
- □ As User I want to have a chat with my events integers, so I can communicate with my team easily.
- □ As organizer I want to organize participant groups, so it allows that they upload the activities or assignments in groups.
- □ As participant I want to see the date and hour of the activities, an how long I have to upload these activities
- □ As Organizer I want to watch if the participants sent the activities at time.
- □ As participant I want to give my opinions about the virtual event, so give my feedback and suggestions to improve the event.
- □ As organizer I want to read the feedback and suggestion of the participants of my events, so I will decide what things I can change.

Entities:

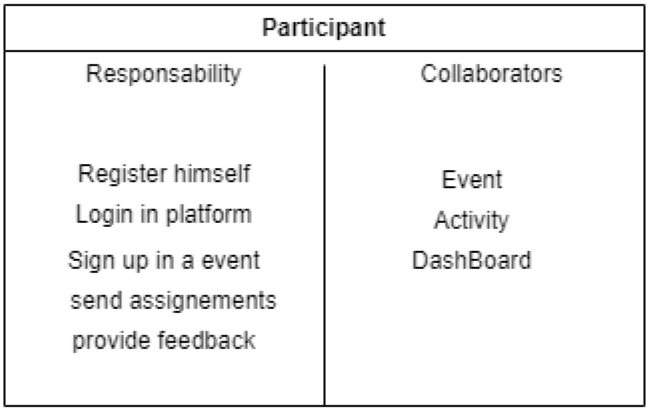
- User
- Account
- Event
- Organizer

- Participant
- Activity
- Assignment
- Type activity
- Activity description
- Documents
- Videos
- Access
- DashBoard
- Chat
- Group
- FeedBack.

Crc cards:

The CRC Cards (Class, responsibility, collaborators) are cards to specify the responsibility of each one and the collaborators that make this possible. Basically specifies the interactions between classes.

The following diagrams are the specification of the responsibilities and the collaborator classes of each main class in the program.



Participant

Responsability	Collaborators
Register himself Login in platform Sign up in a event send assignements provide feedback	Event Activity DashBoard

Organizer

Responsability	Collaborators
Create event	Event
Create activity	Activity
add material	Participant
add participant to an event	
Create participants groups	

Event

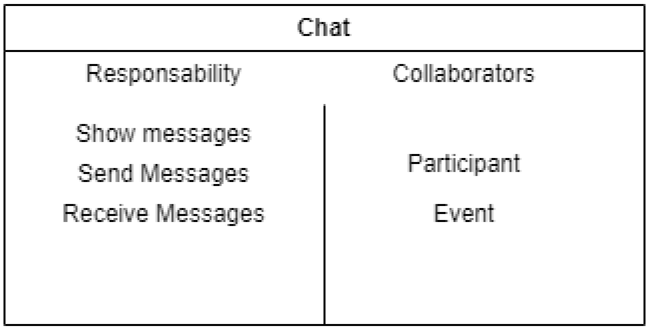
Responsability	Collaborators
	Activity
Show activities	Participant
access chat	organizer
	Chat

Activity

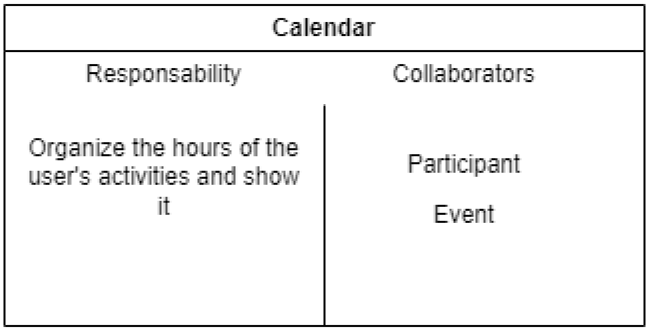
Responsability	Collaborators
Show itself attributes	Event
	Calendar

Dashboard

Responsability	Collaborators
Show the options of the user (Participant or Designer)	Participant
	Event
	Calendar



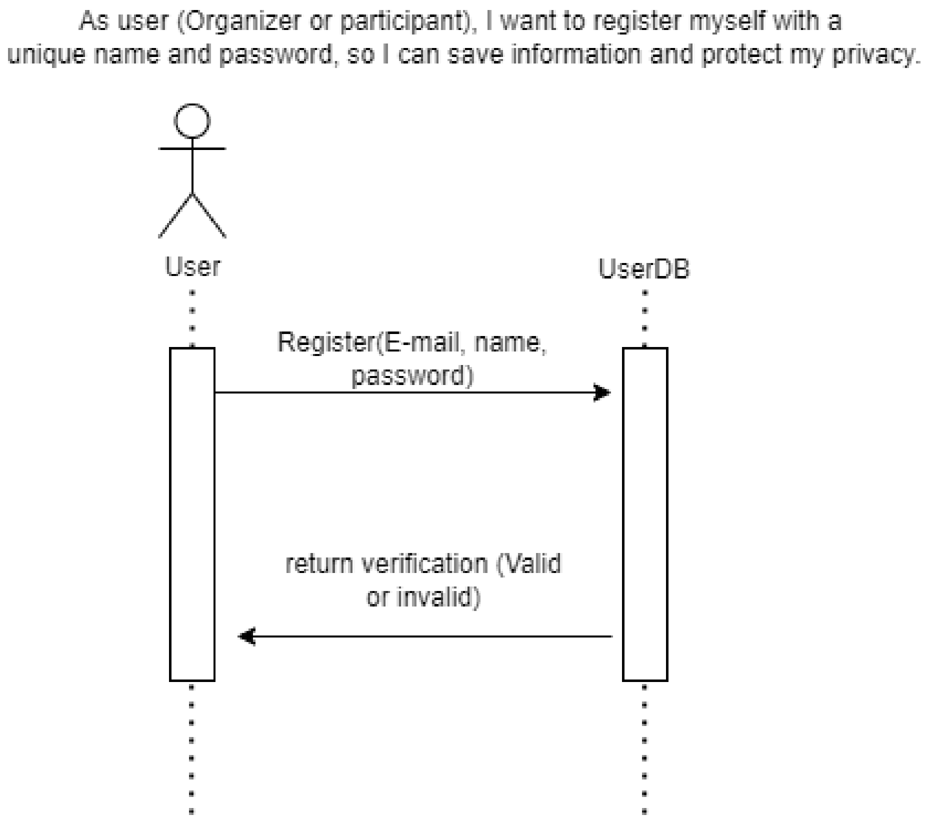
Chat	
Responsability	Collaborators
Show messages Send Messages Receive Messages	Participant Event



Calendar	
Responsability	Collaborators
Organize the hours of the user's activities and show it	Participant Event

Activity diagrams:

Sequence diagrams:

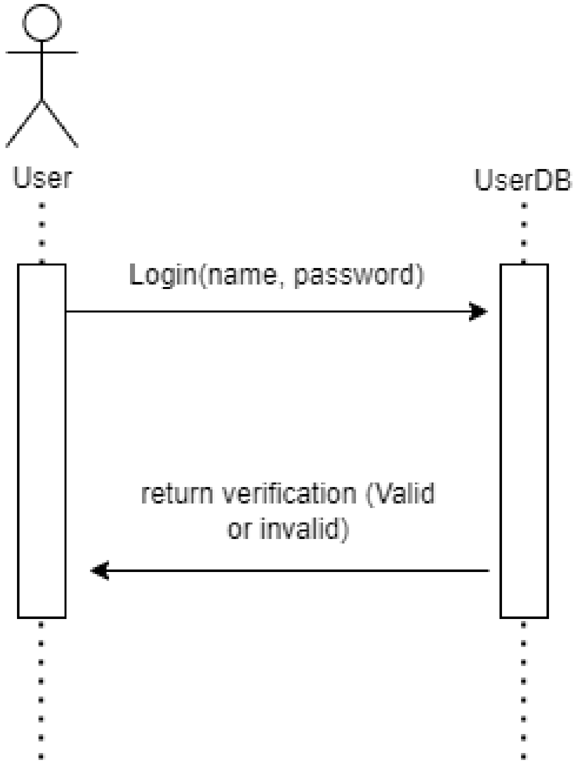


As user (Organizer or participant), I want to register myself with a unique name and password, so I can save information and protect my privacy.

User
UserDB
Register(E-mail, name, password)
return verification (Valid or invalid)

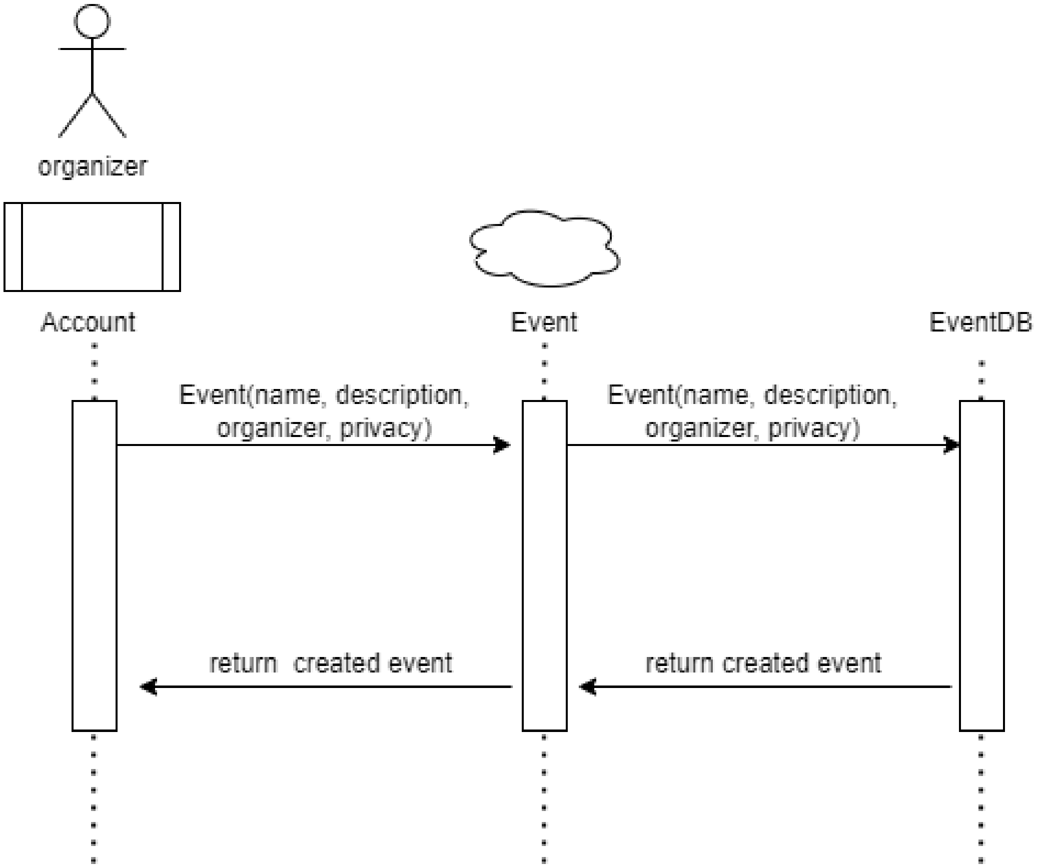
(This diagram shows the user Register with a method in order to get a verification of the Database. The UserDB will return the verification. It will true if the user and e-mail are already registered, and it will false if it not, so the program can guarantee the unique existence of an user.)

As user (Organizer or participant), I want to login with my account, so I can access my information and can know in which events I am organizer or participant.



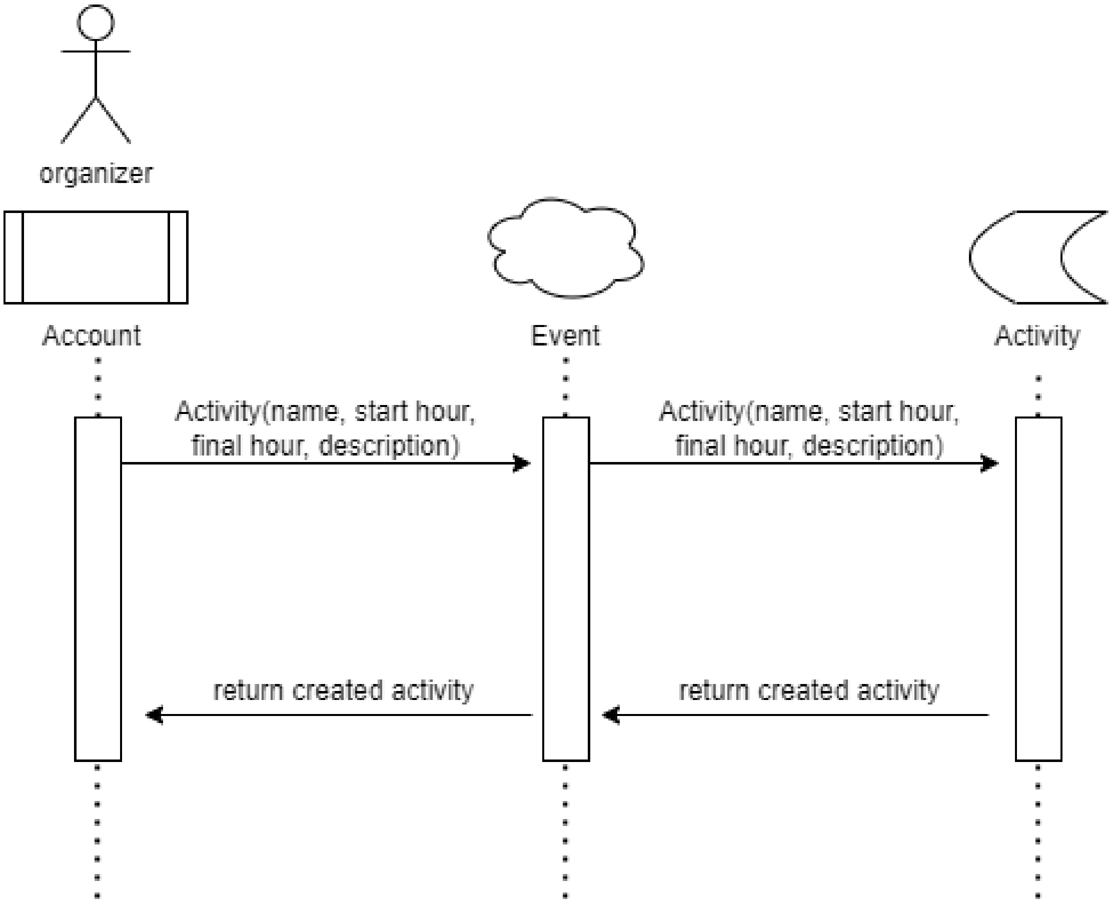
(This diagram shows the user Login with a method in order to get a verification of the Database. The UserDB will return the verification. It will true if the username and password are correct, and it will false if it not, so the program can guarantee the privacy and security of the users.)

As organizer, I want to create virtual events, So I can plan the events in a successful way.


organizer
Account
Event
EventDB
Event(name, description, organizer, privacy)
Event(name, description, organizer, privacy)
return created event
return created event

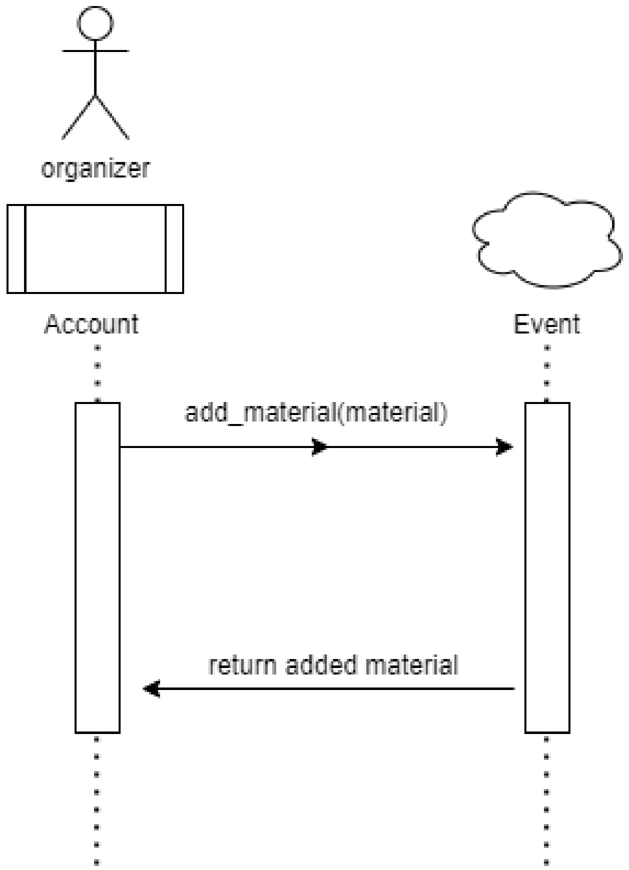
(This diagram shows how an organizer can create virtual events with the method __init__() of the Event class, and it will be added to the Database)

As organizer I want to manage all my virtual events, also add the activities and assignments, so the participants can interact with these activities.


organizer
Account
Event
Activity
Activity(name, start hour, final hour, description)
Activity(name, start hour, final hour, description)
return created activity
return created activity

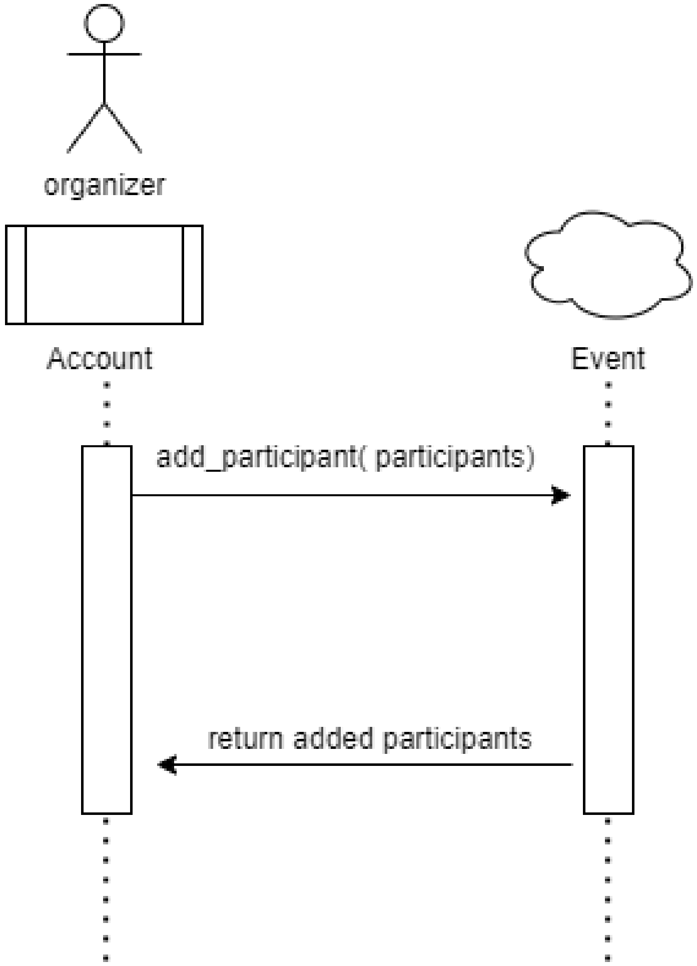
(This diagram shows how an organizer can create activities of the virtual events with the method __init__() of the Activity class, when is created it will be added to the activities_list of an event)

As organizer I want to share documents and videos as material to my participants.



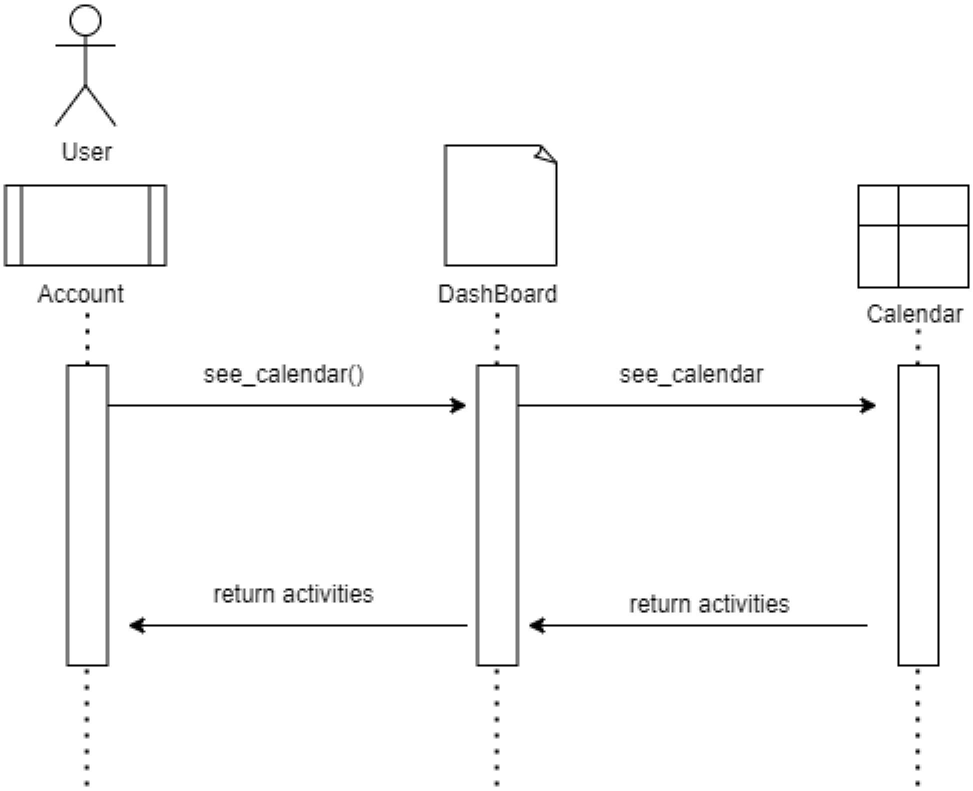
(This diagram shows how an organizer can add support material with the method add_material() so that the participants of an event can access it.)

As Organizer I want to restrict the access to some users, so I can decide who can assist in my events protecting my privacy.



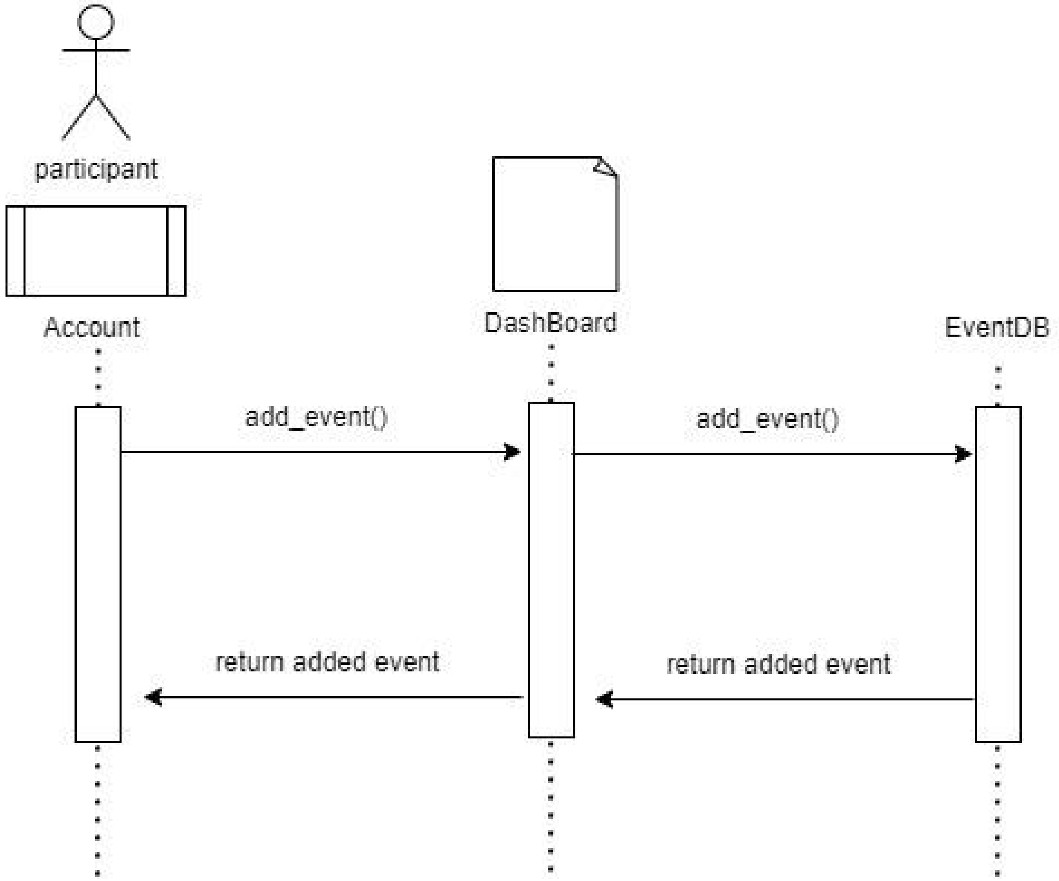
(This diagram shows how the organizer can protect his event privacy restricting the access to not invited users)

As user I want to see my DashBoard, so I can watch my future activities.



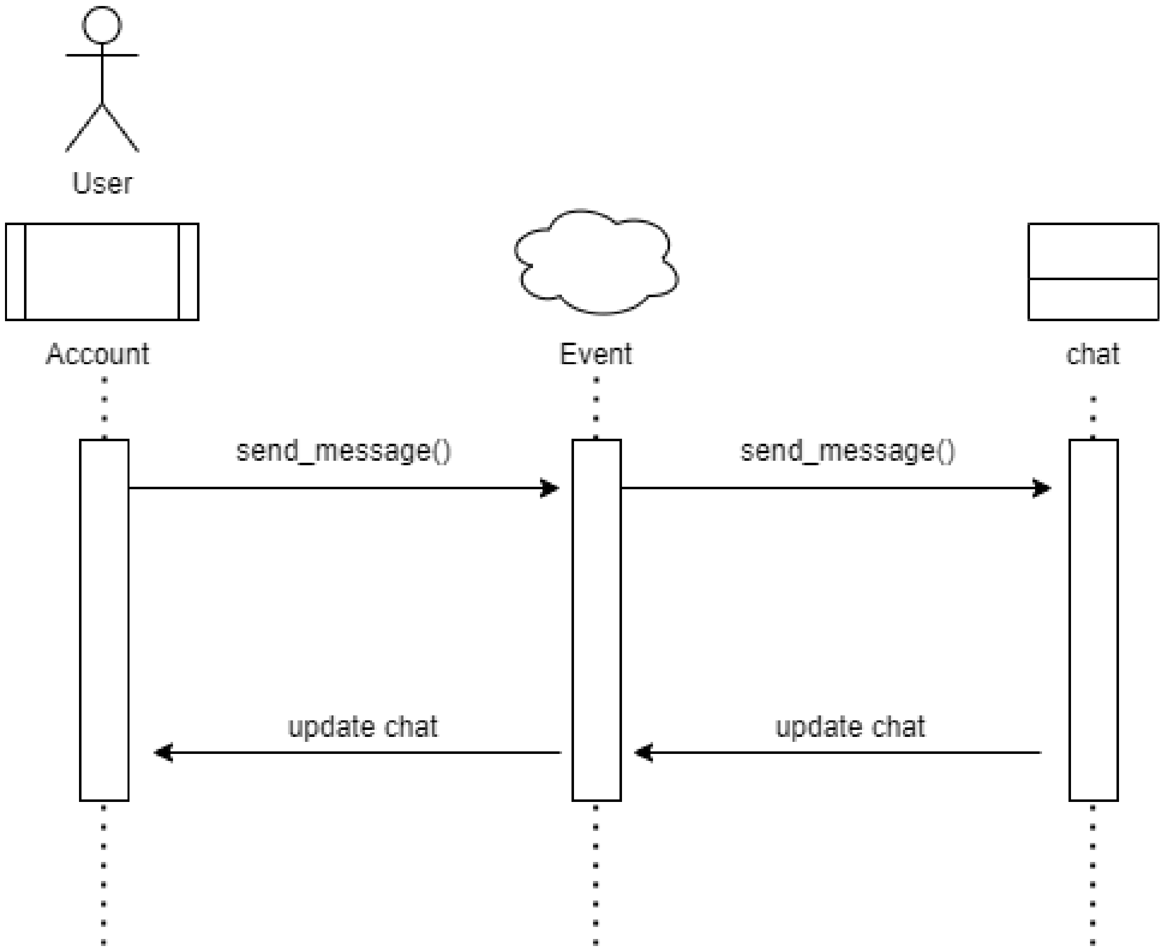
(This diagram shows how the User can access the DashBoard to see his activities in a calendar).

As Participant I want to Sign up to an event.



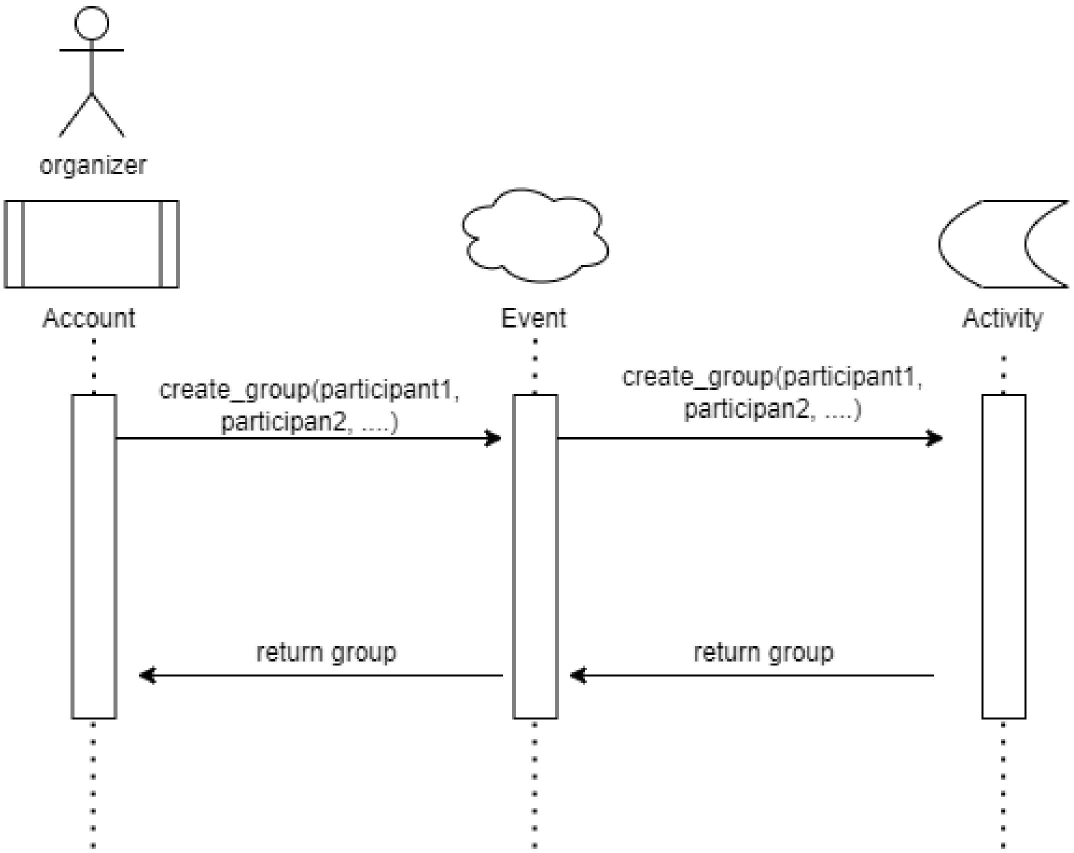
(This diagram shows how a participant can sign up to a event with the add_event() method, he can watch it in the DashBoard, so it can be all the public events in the Database).

As User I want to have a chat with my events integers, so I can communicate with my team easily.



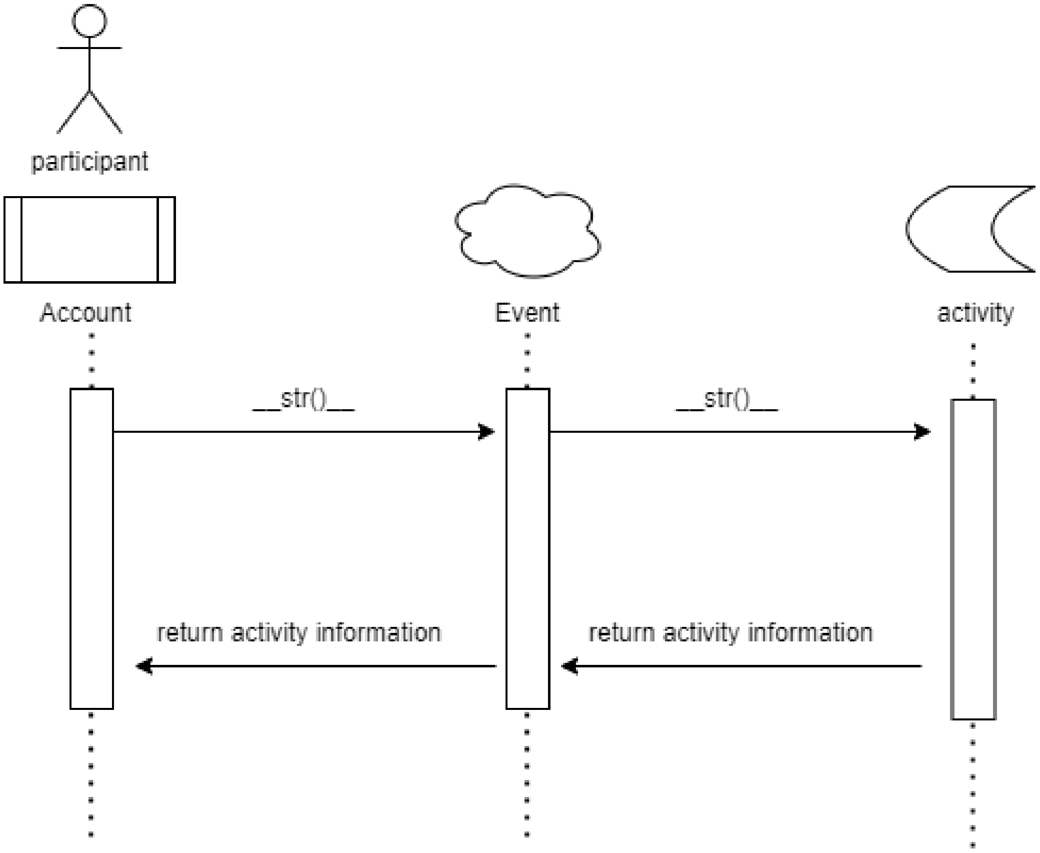
(This diagram shows how an User (Participant or Organizer) can send a message in the event chat, which is once in each event).

As organizer I want to organize participant groups, so it allows that they upload the activities or assignments in groups.


organizer
Account
Event
Activity
create_group(participant1, participan2, ...)
create_group(participant1, participan2, ...)
return group
return group

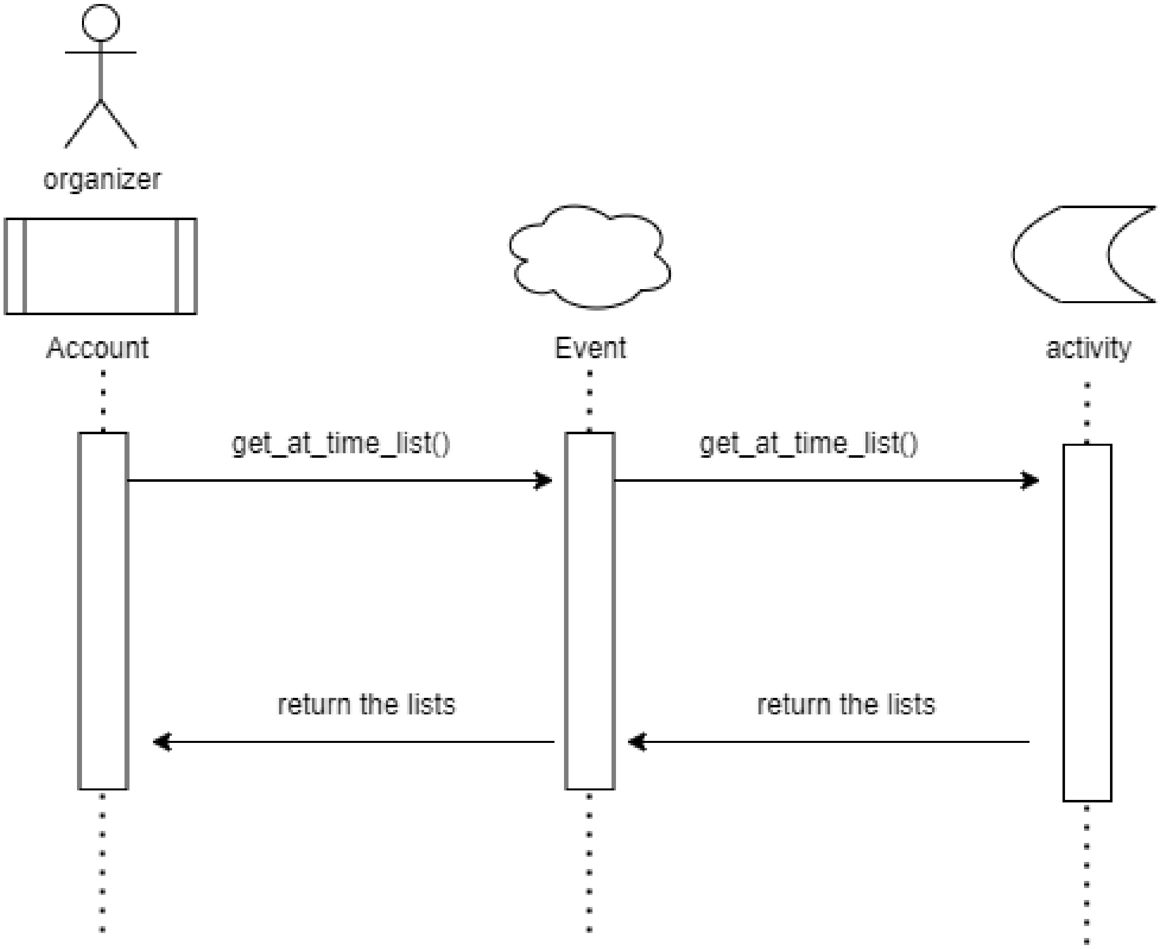
(This diagram shows how the organizer can organize the participants in groups in order for participants to send their activities together).

As participant I want to see the date and hour of the activities, an how long I have to upload these activities



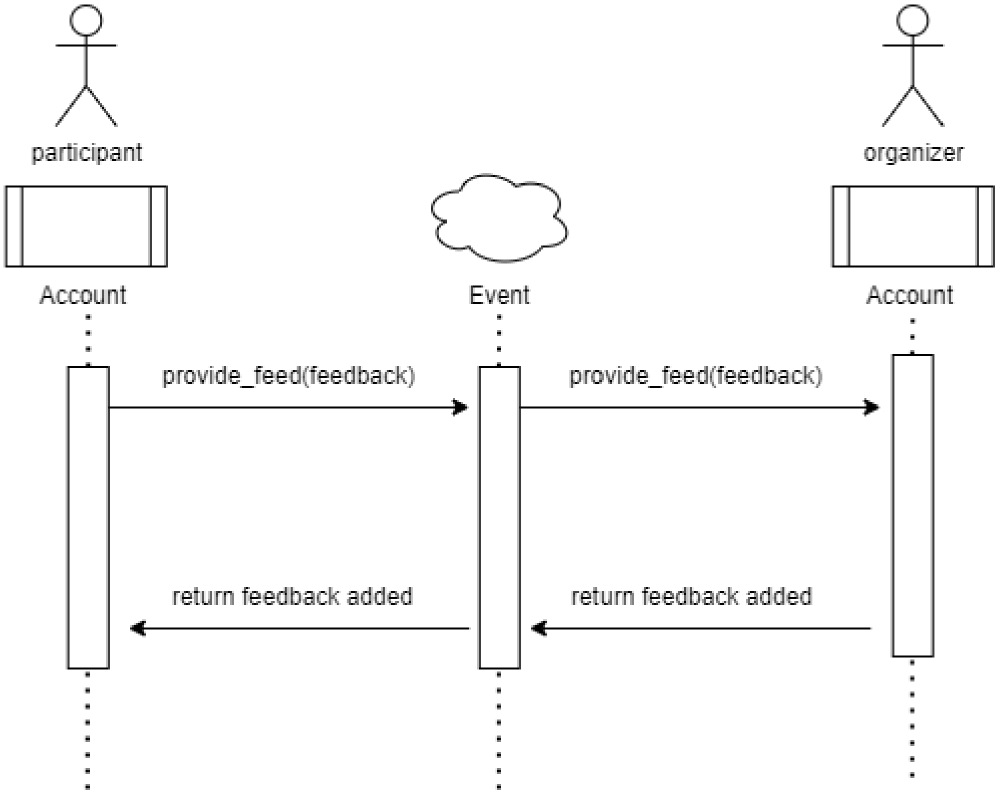
(This diagram shows how the participant can watch the attributes of an activity of an event).

As Organizer I want to watch if the participants sent the activities at time

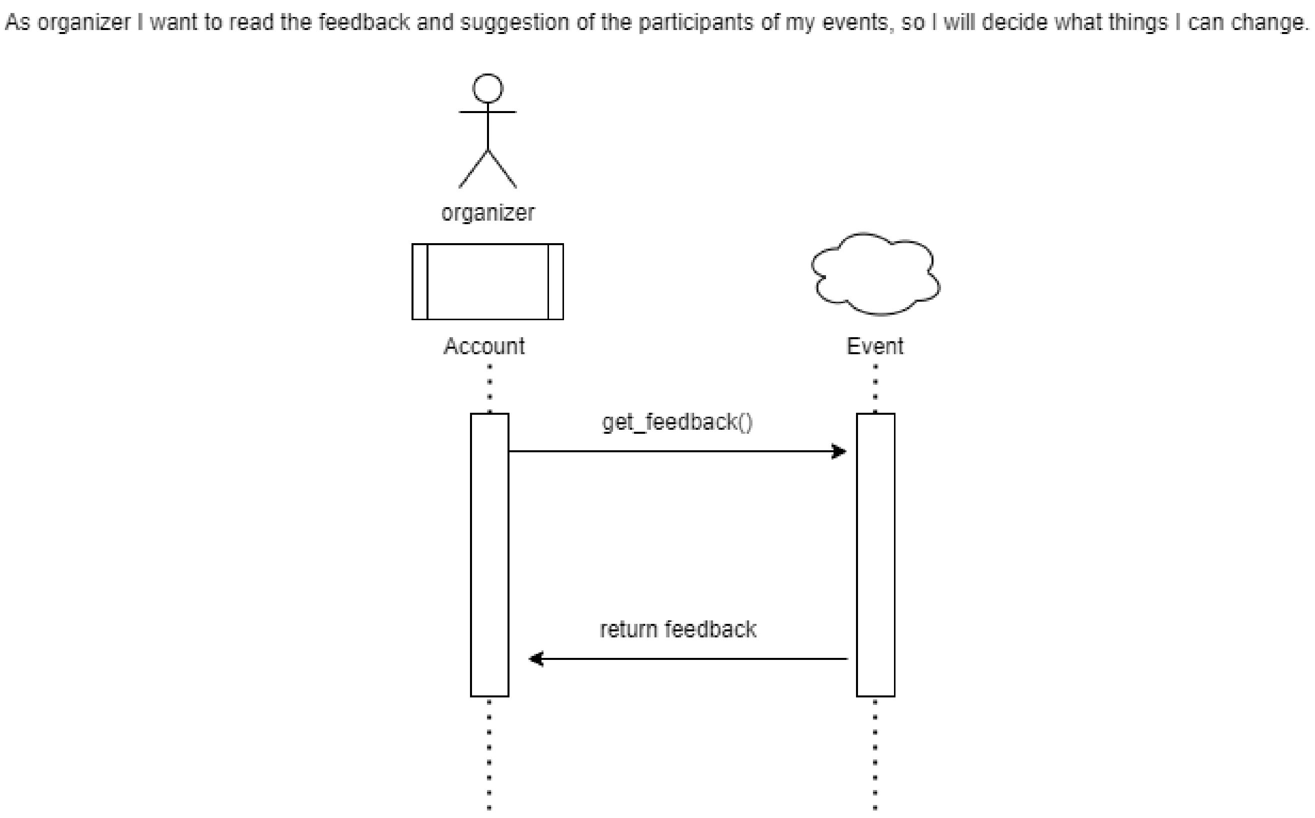


(This diagram shows how an organizer can watch the lists of the participants that send their activities in a limit of time established by the organizer of an activity of an event).

As participant I want to give my opinions about the virtual event, so give my feedback and suggestions to improve the event.



(This diagram shows how a participant can provide a feedback of the event in order to the organizer can watch it and can change things in future events).

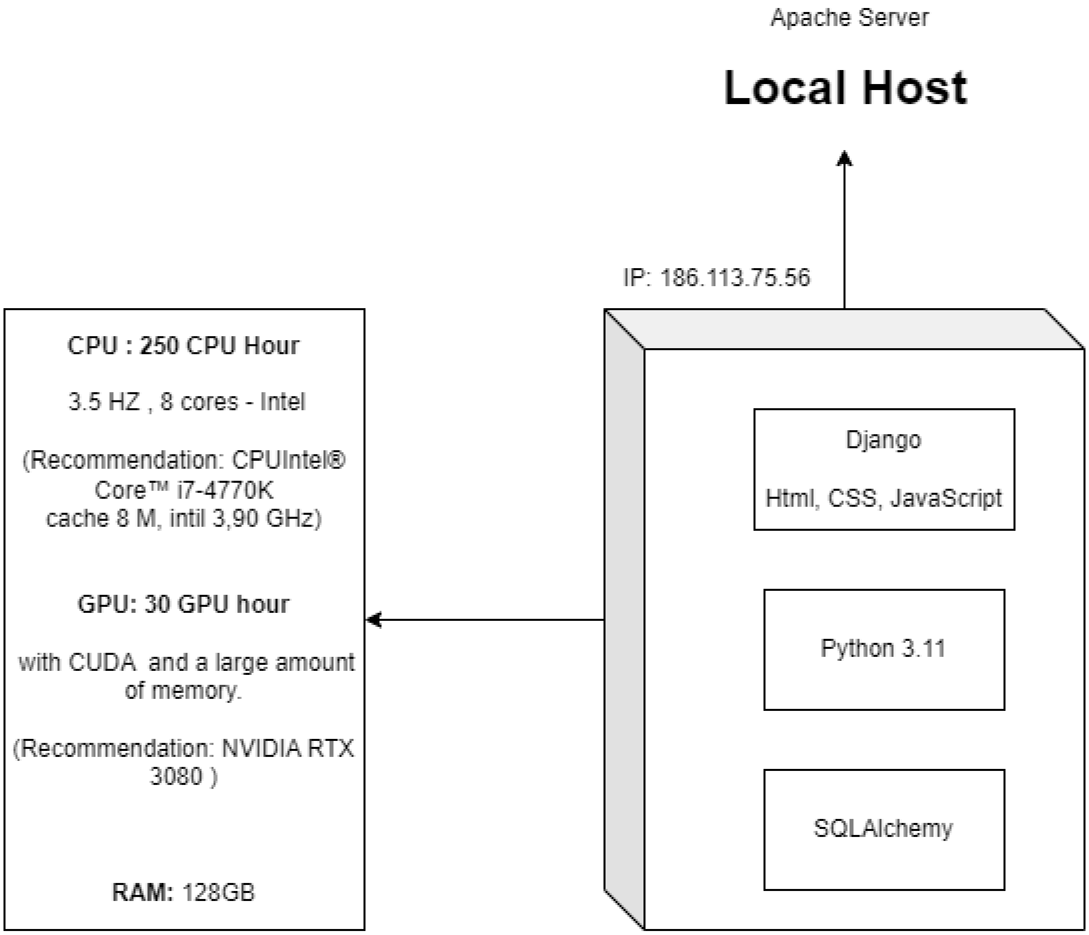


As organizer I want to read the feedback and suggestion of the participants of my events, so I will decide what things I can change.

(This diagram shows how the organizer can read the feedback of the earlier diagram)

State diagrams:

Deployment diagram:



(This diagram shows the physical requirements of the components of the server)

The before data was calculated having this information for the program:

Estimated number of concurrent users (U): 3000

Average number of events per user (E): 10

Average number of activities per event (A): 15

Usage times and expected load during those times (L): Let's assume a 90% peak usage rate.

Any specific performance requirements, such as response times or throughput (T): We can assume a response time of 2 seconds and a throughput of 50 requests per second.

.
So, we can calculate the requirements using the following formula:

Requirements = U * E * A * L * T

Requirements = 3000 * 10 * 15 * 0.9 * (50/3600) (requests per second)

Requirements = 3450000

CPU: With at least 3000 concurrent users, an average of 10 events per user, and 15 activities per event, we can estimate that there will be a total of 3000 x 10 x 15 = 450,000 activities. If we assume that each activity requires a certain amount of CPU time, we can estimate the total CPU time required for all activities. For example, if we assume that each activity requires 1 CPU seconds, then the total CPU time required would be 450,000 CPU seconds.

So, we estimate 250 CPUs hours,

GPU: To determine the GPU requirements, we need to consider the graphics and processing, how it will be programmed in python, css and html. So a common recommendation for GPU requirements is to have at least 1 GPU per 100 users. Therefore, for 3000 users, we would need at least 3000 / 100 = 30 GPUs.

RAM: A common recommendation for RAM requirements is to have at least 1 GB of RAM per CPU. Therefore, for 250 CPUs, we would need at least 250 GB of RAM.But how this is the first version of the application, we will use 128 GB

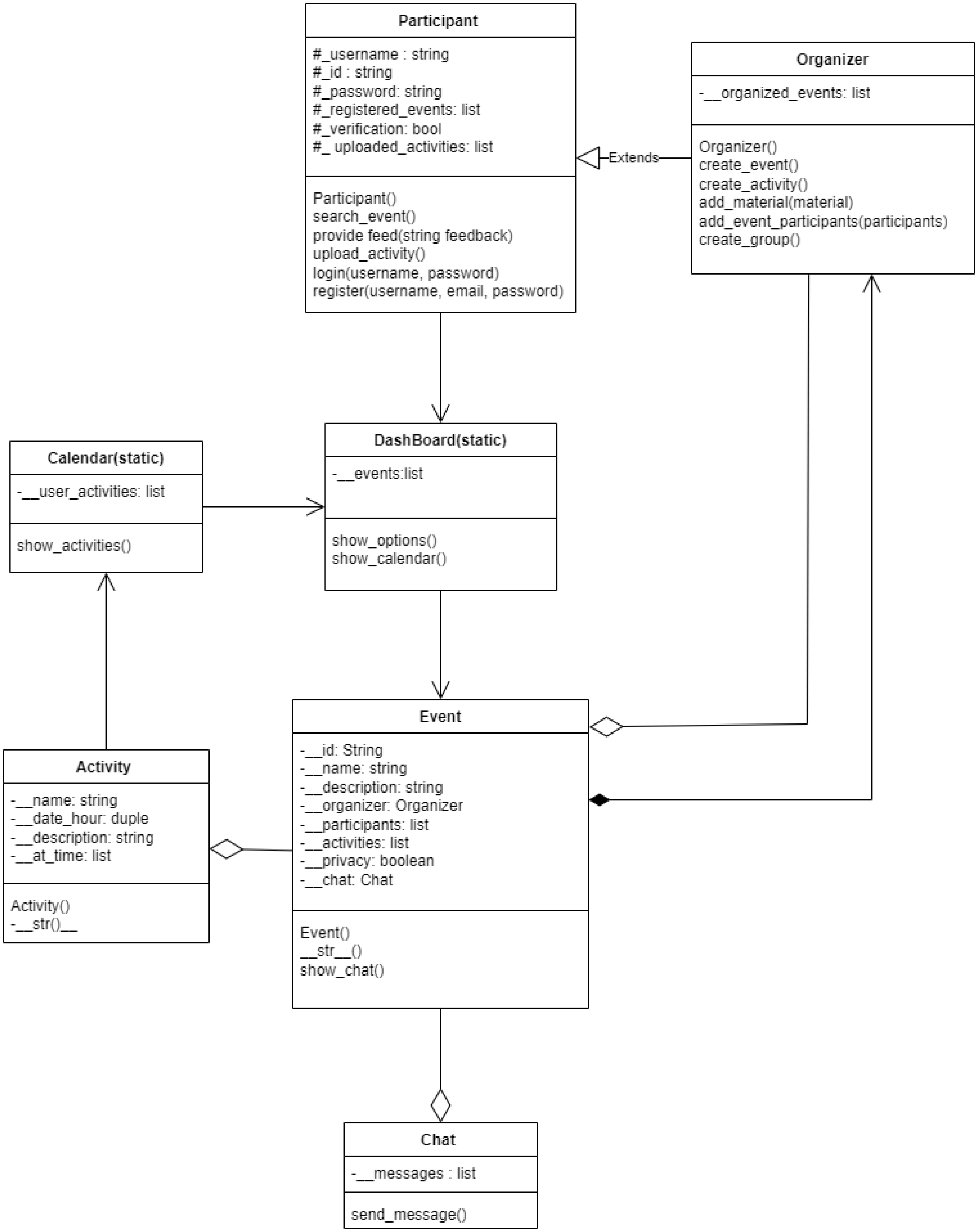
So, we found in the market some components that can support these requirements..

For CPU, we estimate requirements, we would recommend considering a server with a multi-core CPU, such as an Intel or AMD processor, with at least 8 cores and a clock speed of 2.5-3.5 Ghz. This would provide sufficient processing power to handle multiple simultaneous events and activities, so, we recommend CPUIntel® Core™ i7-4770K with cache 8 M, until 3,90 GHz.

We recommend a GPU with CUDA to manage it and a large amount of memory like NVIDIA RTX 3080.

And for the RAM, can be any memory with 128GB of RAM.

Class diagram:



(This diagram shows how the classes interact one each other)