Sequence Diagram – Manage an Event

This sequence diagram illustrates the interactions involved when a venue or a verified user manages events through creation, editing, or deletion.

Participants

- **User** – The end user who initiates event management actions.
- **EventManagementDialog** – The interface through which the user manages events.
- **PopUpDialog** – A confirmation dialog used during deletion.
- **EventController** – The logic controller that handles event operations.
- **ProxyDbController** – The controller responsible for data access.
- **EventEntity** – The event data model being manipulated.

Scenarios and Flow

Scenario 1: Create Event

- The user initiates the creation process via `clickCreateEvent()`.
- `EventManagementDialog` calls `createEvent(details)` on the `EventController`.
- `EventController` passes the details to `ProxyDbController`, which invokes `create()` on `EventEntity`.
- The created `Event` is returned back through the chain and displayed using `displayEvent()`.

Scenario 2: Edit Event

- The user selects to edit an event via `clickEditEvent()`.
- `EventManagementDialog` invokes `editEvent(details)` on the `EventController`.
- `EventController` forwards the request to `ProxyDbController`, which updates the event by creating a new `EventEntity`.
- The updated `Event` is returned and displayed via `displayEvent()` to the user.

Scenario 3: Delete Event

- The user initiates deletion with `clickDeleteEvent(eventName)`.
- `EventManagementDialog` opens a confirmation popup by calling `popDialog()` on `PopUpDialog`.
- An `alt` block captures two outcomes:
  - If the user **confirms deletion**, `deleteEvent()` is called again, and the `EventController` processes the deletion via `deleteEvent(details)` in `ProxyDbController`.

- If the user **declines**, the process ends and readresses the user to the eventManagementDialog.