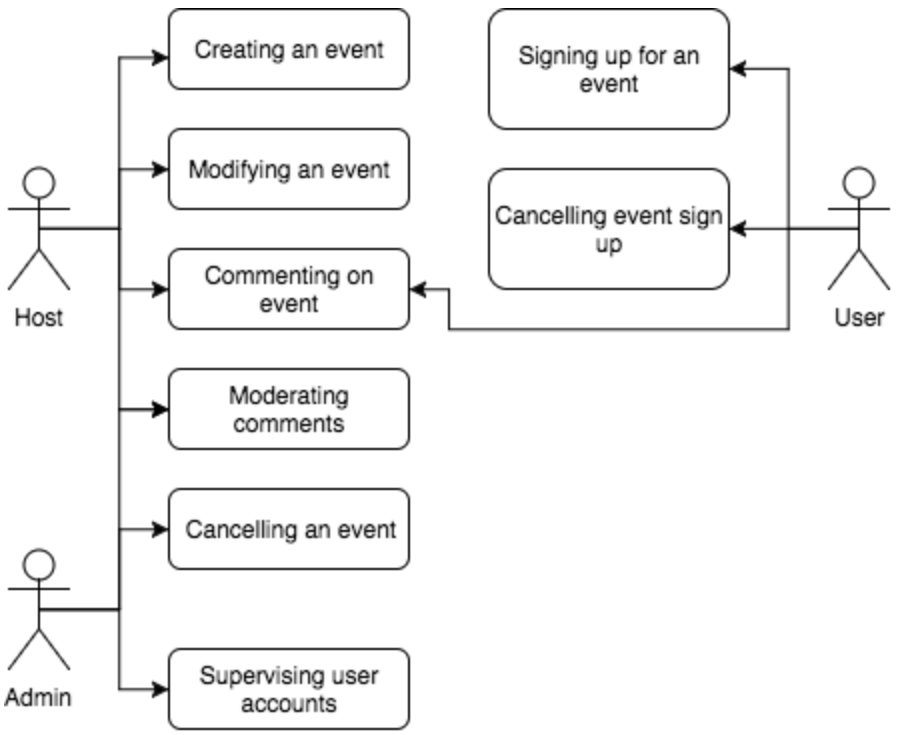
HQ-Calendar

Preface

HQ-calendar is a calendar/scheduling application which allows users to conveniently create events, sign up for events and comment on events and provides hosts and administrators with tools to manage the application.
The application back end is a RESTful API made with Node.js using Express web framework and PostgreSQL as the database. The front end is made with React which is a modern JavaScript library developed by Facebook. Both front and back end are run in an UNIX server with Nginx used as http server for front end and reverse proxy for back end.

Use case



User groups

User is any registered user.
Host is an user who has created an event or has been assigned as a host for an event.
Admin is a superuser appointed by application owners.

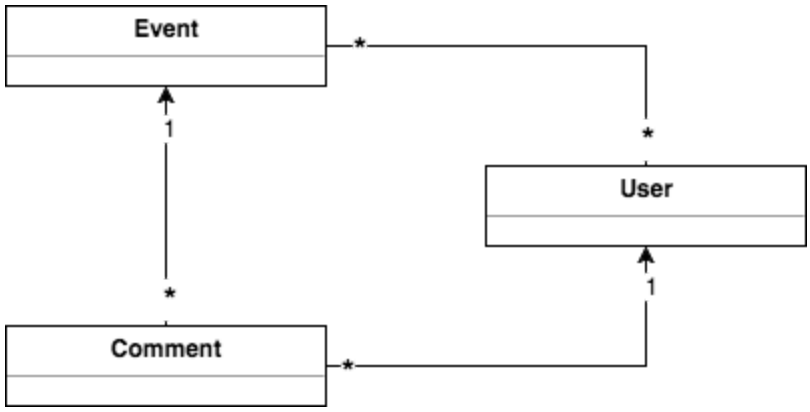
Use cases

User use cases
Signing up for an event:
anyone can sign up for an event signifying they are attending the event
Cancelling event signup:
Users can withdraw their signup from events they signed up for
Commenting on events:
Depending on event anyone or event attendees can post comments on the event page.

Host use cases
Creating an event:
Anyone can create an event. Doing so they become the host of the event and can add other hosts and modify event information.
Cancelling an event:
Hosts can cancel an event they have created.
Moderating event comments:
Hosts can delete comments from their events.

Admin use cases
Managing accounts:
Admins can modify and ban users from the app or disable their event creating privileges.
Everything else:
Admins are superusers and can do every possible action on the app.

Database diagrams



Event

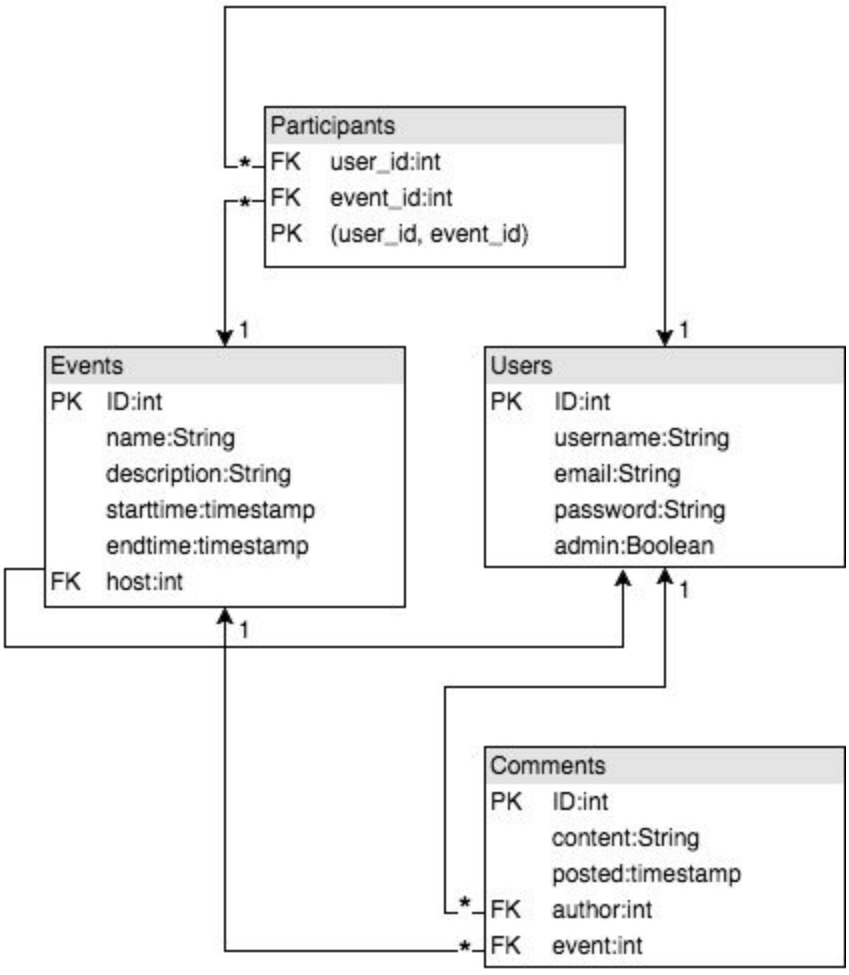
Attribute	Type	Description
Name	String, max. 50 characters	Name of the event
Description	String	Information about the event
Starttime	Timestamp	When the event starts
Endtime	Timestamp	When the event ends
Host	Integer	User who is hosting the event

User

Attribute	Type	Description
Username	String, max. 30 characters	Name of the user
Email	String, max. 50 characters, unique	Users email address
Password	String	Salted password hash
Admin	Boolean	Indicate if the user is admin

Comment

Attribute	Type	Description
Author	Integer	Who wrote the comment
Event	Integer	What event is the comment in
Posted	Timestamp	When the comment was posted
Content	String	Content of the comment


Participants
FK user_id:int
FK event_id:int
PK (user_id, event_id)
Events
PK ID:int
name:String
description:String
starttime:timestamp
endtime:timestamp
FK host:int
Users
PK ID:int
username:String
email:String
password:String
admin:Boolean
Comments
PK ID:int
content:String
posted:timestamp
FK author:int
FK event:int
1
*

Application composition

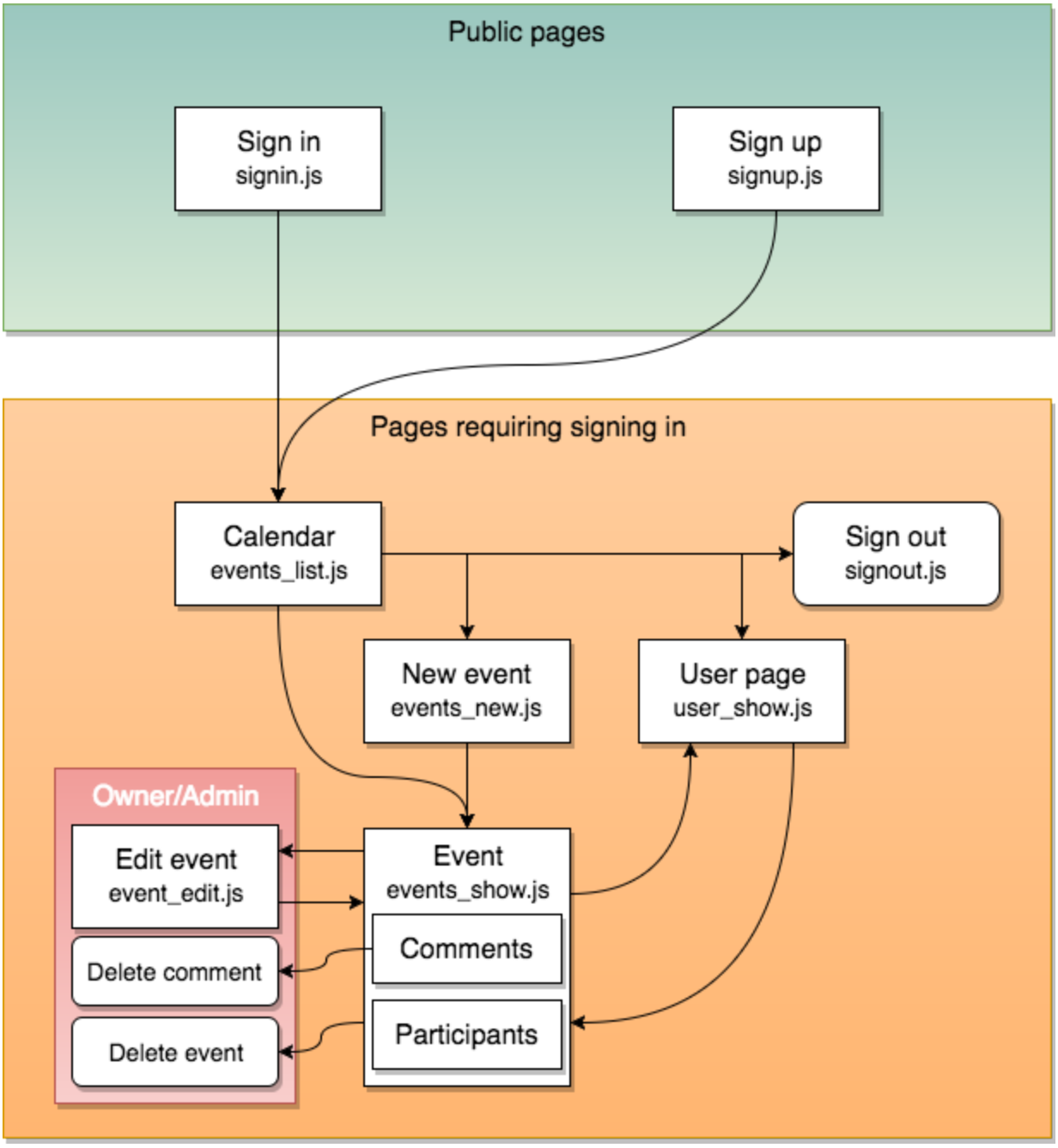
Application follows MVC-model flexibly. Because backend and frontend are separate applications there are some anomalies. Models and controllers are found in `server/src/` directory and views are found in `client/src/` directory.

In backend model and controller files are located in their respective directories and named accordingly. Participation is a cross reference table and its controllers and models are located in event table's files. Backend authentication logic is located in `server/src/services` directory and user controller. Backend provides the frontend with a JWT token to use for authentication.

Frontend is built with React and Redux and the "dumb components" which are views not connected to Redux store are located in `client/src/components`. "Smart components" are views that are connected to Redux store and they are located in `client/src/containers` and divided in events, users and auth. Redux action creators are located in `client/src/actions` and action reducers in `client/src/reducers`, both in their own files respectively. Frontend uses the browser's localStorage to store the authentication token.

Node modules which are the helper libraries are listed in both server and client directory in `package.json` file. They are installed in `node_modules` directory when running `npm install`. Database settings are in `server/src/db/database.js` file which reads some of them from environment variables. JWT secret is read from `server/src/config.js` file, there's an `exampleconfig.js` file in the repository.

UI and components



Instructions

After logging in user can create new events from the *New Event* page or find an existing event to join from the *Events* page. Editing or deleting events or comments is possible for the event host from the event page. Users can delete their own comments by hovering over their comment to reveal the delete button.

Creating an account:

1. Input your information in the signup page and press Sign Up

Signing in:

1. Enter your credentials and press Sign In

Signing out:

1. From the navigation bar, open the dropdown menu by clicking your username and press Sign Out

Creating a new event:

1. Navigate to New Event page from the navigation bar link or from url `/events/new/`
2. Enter at least name and time for the event, ending time can't be before starting time.
3. Press Create to create your event or Cancel to abort

Editing an event:

1. Navigate to the events page, you can find your own events from your user page
2. Press Edit, make the changes you wish to edit or select the comments you wish to delete and press Save Changes or Cancel to abort

Deleting an event:

1. Navigate to the events page, you can find your own events from your user page
2. Press Delete and confirm deletion in the modal popup

Posting a comment:

1. On the event's page, type your comment in the comment field and press enter or send

Deleting a comment:

1. On the event's page, hover your cursor over your comment you wish to delete and press the trash bin icon.

Installation

- Download/clone the repository
- Run `npm install` in **both** `server` and `client` directories
- Configuring the database
    - Create PostgreSQL database “calendar”
    - Create the tables with `psql calendar < server/sql/init.sql`
    - Setup your database address, port, username, password and name if you didn’t use “calendar” in `server/db/database.js` and/or in environment variables.
    - Add some test data to the database with `psql calendar < server/sql/seed.sql`
- Configuring the server
    - Rename the exampleconfig.js to config.js and change the secret phrase
- Run `npm start` in **both** `server` and `client` directories
- The server is now running in `localhost:3001` and the client in `localhost:3000`

Deployment

- Run `npm build` in `client` directory
- Move, link or serve only the `index.html` file from `client/build` directory with NGINX, React handles the 404
  `location / { try_files $uri $uri/ /index.html; }`

- Configure your NGINX to serve the server with proxy pass at location /api/
  `location /api/ { proxy_pass http://localhost:3001; }`