Trackitime

# What?

How many hours did you use with your practice work? Or with your origami project? Trackitime is an application to solve this problem. With Trackitime one can easily create a new project, and add time periods spent with the project. Later some statistics can be seen of the project: what have been done, how much time have been spent altogether and so on.

# how?

Trackitime is used with an easily-used web application, which is also supportive for mobile browsers. The users are identified with a logging system and personal accounts.

# With what?

The language Trackitime is build with is JavaScript with [Node.js](https://nodejs.org/). Libraries used are [Express.js](http://expressjs.com/) (with all its dependencies) for creating and running the server and routing the requests. The user system is implemented in backend with [Passport.js](http://passportjs.org/). All data is saved into a PostgreSQL database.

# where?

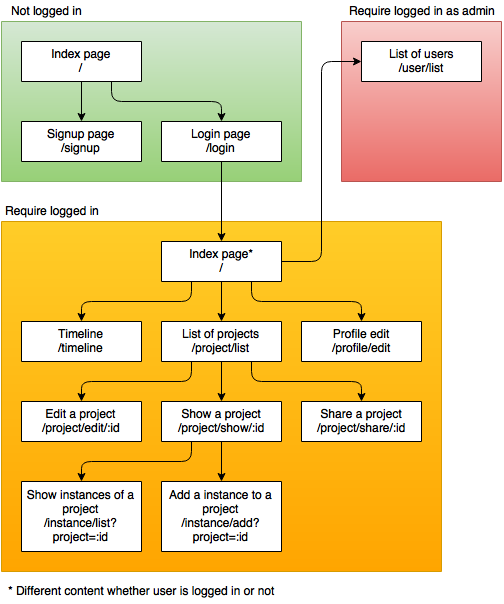
The code of Trackitime is stored in Git with a repository of the same name. The address of the repository is <https://github.com/Aapzu/trackitime>. At this moment the app is running in Heroku, and it can be found from the address <https://trackitime.herokuapp.com/>.

structure of the application

# Files and folder

The application follows the MVC model. Models are found from /models, views from /views and controllers from /routes. Configuration files/scripts are in folder /config, and other backend scripts in /app. All front-end related (except the views themselves) is found from /public.

# system components



Use cases and users

# users

## Unregistered user

## Registered user

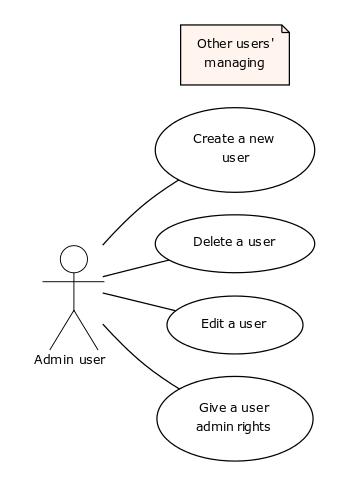
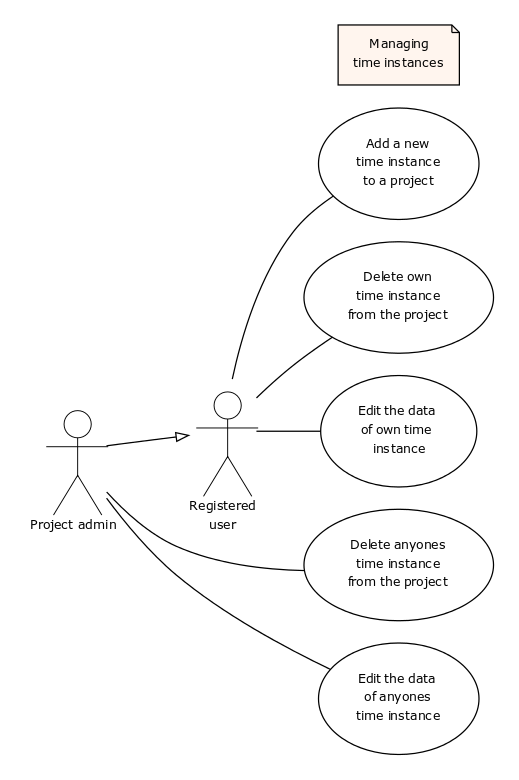
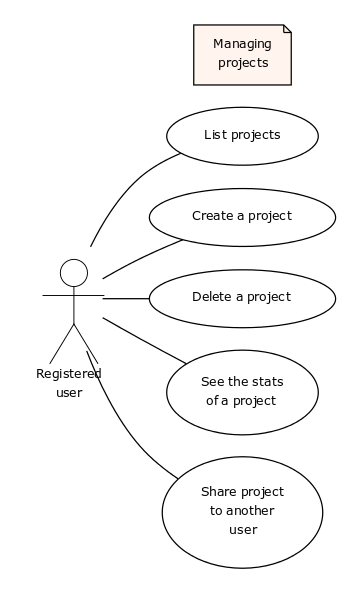
A registered user has signed up for the app and is logged in with their account. Almost all of the use cases require being logged in.

## Admin user

Has all the rights of a registered user, but can also remove/edit other users.

# use cases

## Diagrams



## Explanation for the most important use cases

## Unregistered user

### Sign up

By signing up an unregistered user creates an account for themselves. The real name, username and password are required for signing up.

## Registered user

### Create a project

A registered user can create projects. Projects can/must have some data, e.g. name, description and the date started. All time instances must belong to at least one project

### Add a time instance to a project

A time instance must have a date performed, description, duration and a project to belong to.

### See the stats of a project

All the time instances used to a project can be seen from the statistics page of the project. The page can show at least the total amount of instances used, the total amount of time used, the average time spent per instance and so on.

Data

# Data content

# tables

## User

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Description | Foreign key? |
| id | Integer | Auto incremented id |  |
| name | String, max 50 char | Real name of a user |  |
| username | String, max 50 char | Unique username, cannot be empty |  |
| password | String, max 50 char | Hashed password |  |

## UserProject [connection table]

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Description | Foreign key? |
| user | Integer |  | User.id |
| project | Integer |  | Project.id |
| isAdmin | Boolean | Is the user the admin of the project |  |

## Project

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Description | Foreign key? |
| id | Integer | Auto incremented id |  |
| name | String, max 50 char | Name of the project |  |
| description | String, max 500 char | Description of the project |  |
| date\_started | Date | Date project is started |  |

## TimeInstance

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Description | Foreign key? |
| id | Integer | Auto incremented id |  |
| description | String, max 500 char | Description of the time instance |  |
| from | DateTime | DateTime when started |  |
| to | DateTime | DateTime when stopped |  |
| project | Integer | The project the instance belongs to | Project.id |
| user | Integer | The user who performed the time instance | User.id |

# Database diagram

installation and usage

# installation locally

* git clone https://github.com/Aapzu/trackitime
* Go to the cloned directory
* Create a PostgreSQL database
* Create the tables by running sql/create\_tables.sql to the database
* npm install
* DATABASE\_URL=<FULL URL> [PORT=<PORT>] npm start where <FULL URL> is the PostreSQL url containing host, port, username and password and <PORT> the port app starts in (optional, default 8080)
* Go to [http://localhost:8080](http://localhost:8080/) or [http://localhost:<PORT](http://localhost:%3CPORT)>
* sql/add\_test\_data.sql
* You can also run sql/add\_test\_data.sql to the database. That creates two test users and a couple of projects and time instances to the database. Otherwise one can always create new users by signing up to the system (but not administrators).
  + Users:

| **Username** | **Password** | **Full Name** | **User Rights** |
| --- | --- | --- | --- |
| admin | admin | Admin Tester | Administrator |
| tester | tester | Basic Tester | User |

Username ‘tester’ also works in the published version in Heroku.