FOSS implementation of the WorkAdventure Admin Services

Tobias Tefke <t.tefke@stud.fh-sm.de>

September 20, 2021

1 Introduction

Workadventure[1] is a web application which features online meetings. It differs from other online conferencing solutions by its design. The applications resembles a game in eight bit optics. Participants have avatars which can move on a two-dimensional map. If characters get close to each other they can talk to each other. The talk can be left by moving away. Furthermore, it is possible to embed special actions on the map tiles. This includes opening websites and starting Jitsi conferences[2].

2 Motivation

An open source version of WorkAdventure can be self-hosted[3]. However, a paid version with additional features is available. These features include sending global messages and access control. If someone decides to use these functions, a proprietary administration component has to be included. Furthermore, a fee based on the number of simultaneous connections must be paid[4]. Moreover, user-specific data is being exchanged with the administration component. This makes it hard to integrate WorkAdventure into environments in which user data must be well protected, e.g. education institutions. In order to mitigate the mentioned problems, it was decided to re-implement the administration services as open-source software.

[1] `https://workadventu.re` (acc. 09/19/2021)
[2] `https://workadventu.re/map-building/opening-a-website.md` and `https://workadventu.re/map-building/meeting-rooms.md` (acc. 09/19/2021)
[3] `https://github.com/thecodingmachine/workadventure` (acc. 09/19/2021)
[4] `https://workadventu.re/pricing` (acc. 09/19/2021)

3 Background

The communication messages sent between the proprietary and open-source service are defined in the pusher and back-components[5]. The open source components request services from the admin services by sending HTTP requests using axios[5][6]. Here, the WorkAdventure services have to authenticate by sending an authentication token[5]. If the authentication succeeds, the admin API responds with a JavaScript object containing the requested information, serialized in JSON[5].

4 Implementation

Using the knowledge about how communication works, it is possible to reimplement the proprietary services. A coarse overview about the reimplementation is shown in Figure 1. The HTTP requests are sent to the URL of the admin services. This URL is set to our admin service implementation. There, an nginx server handles the requests and the requested PHP scripts are being called. PHP reads the necessary data to respond from an SQL database and returns the expected answer encoded in JSON. The encoded data is then being returned to WorkAdventure, where it can be further processed.
Furthermore, a Website is being hosted on the nginx server, making it possible for the administrator to perform CRUD operations on the information stored in the database using a graphical user interface.

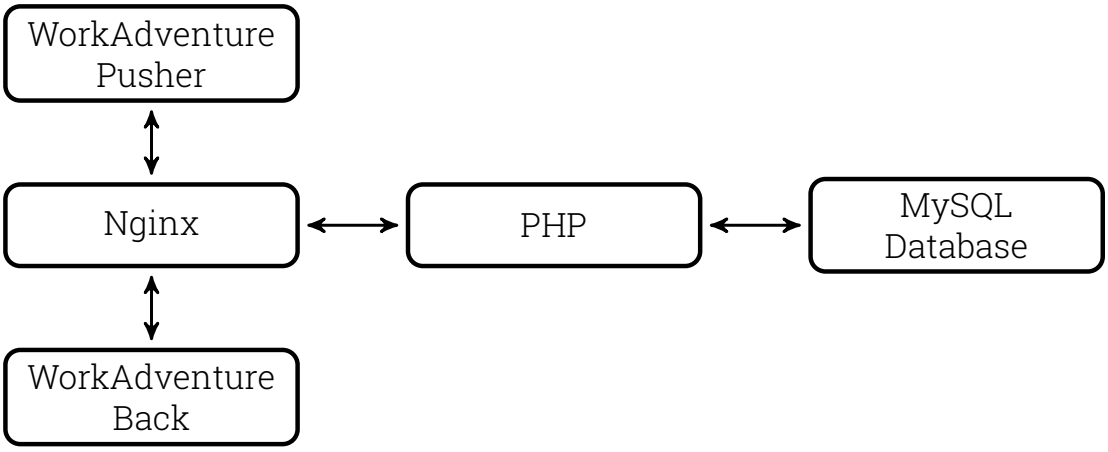

Figure 1: Structural overview about how the administrative components interact with each other.

[5] `https://github.com/thecodingmachine/workadventure/blob/develop/pusher/src/Services/AdminApi.ts` and `https://github.com/thecodingmachine/workadventure/blob/develop/back/src/Services/AdminApi.ts` (acc. 09/19/2021)
[6] `https://www.npmjs.com/package/axios` (acc. 09/19/2021)