

PROJECT TRCKR

TECHNICAL ARTICLE

IT15_{TA_ZH} - PSIT4

Ankeshian, Gabriel

ankesgab@students.zhaw.ch

Balidis, Dimitri

baliddim@students.zhaw.ch

Christen, Luca

chrisluc@students.zhaw.ch

Jossi, Savino

jossisav@students.zhaw.ch

Milenkovic, Daniel

milendan@students.zhaw.ch

Nominato, Angelica Helena Moreira Alves

moreiane@students.zhaw.ch

Pacassi Torrico, David

pacasdav@students.zhaw.ch

23.05.2018

The main goal of the present article is to describe the idea, goals and main functionalities of the web based application trckr. We developed trckr for everyone who works on a project and needs an intuitive and simple web tool to accurately track their time spent on different tasks.

The backend is written in Python with the help of the web application framework Django and the frontend with the Javascript UI framework Vue.js. Both technologies were new to most team members, but all members have proven themselves self-driven and learning these skills was ultimately a benefit to the outcome of the project.

In order to compete with similar tools and web services, trckr focuses on performance and usability. To distinguish trckr from the competition, many features are planned to manage projects and tasks in a user-friendly manner. This will allow the user to leverage trckr to handle the ever increasing complexity in project management and task tracking found in large companies. Despite targeting large companies, trckr will remain open source and anyone can contribute, covering cases we might have never dreamed of.

1 Introduction

Time and task tracking are important activities in many businesses to gain insights on the productivity of a team, requiring appropriate tools allowing easier and more accurate time tracking. Many processes and methods have been developed in the past to cover this need. Unfortunately most of them only address a certain need or have been fine tuned to a specific company or team. This not only leads to a loss of experience that could have been leveraged by other teams but is also not generic enough to adapt to different processes, causing other companies to inappropriately adapt to the tool instead of the tool adapting to the company.

1.1 Objectives

The goal with trckr is to develop and distribute a time tracking web application that is easy to both understand and use. The most important non-functional requirement is to focus on a small amount of steps for users to track all the required information. Only this will keep the user engaged and increase the accuracy of the provided data.

1.2 Main Features

The user is able to:

- register and login to trckr
- create and edit projects
- create, track and edit tasks
- visit trckr on any device

2 Architecture

The trckr project consists of two separate applications. The business and data layer are handled by the backend application, while the presentation layer is implemented through the frontend application. The backend provides a RESTful API, which is used by the frontend application to read and write the necessary data.

This separation of concerns, achieved through the clear decoupling between the frontend and backend, allows for easy extensibility and fast development cycles. Additionally one backend is able to serve multiple client applications, without any further work required.

3 Technologies

3.1 Django

Django is an open source web framework written in Python.[1] It encourages fast, clean and simple development of web applications. One of the main advantages is its fast setup, enabling the developer to create applications swiftly through it's Model-View-Presenter scheme. Django also comes with support for various databases. Many users compare Django to Ruby On Rails but written in Python. Django also follows the DRY principle (Don't Repeat Yourself).

3.2 PostgreSQL

PostgreSQL is an open source object-relational database.[2] The database is used to store all the data for trckr. PostgreSQL is fully supported by Django and requires minimal configuration. The communication between the web application and the database is done through Django's model layer.

3.3 Docker

Docker is a containerization platform, which adds an abstraction layer between the application environment and the underlying server infrastructure.[3] This means that the whole backend environment can be setup inside multiple docker containers, that can be run locally as well as on the server.

3.4 Node.js

Node.js is a Javascript runtime, which brings Javascript to the server environment.[4] It uses an event-driven and non-blocking I/O model for maximum efficiency, while still remaining lightweight. This makes node.js the ideal technology to implement the server side of the trckr frontend application.

3.5 Vue.js

Vue.js is a progressive framework for building user interfaces.[5] Vue.js is used mainly for its simplicity and the rather shallow learning curve it provides inexperienced developers. The features of Vue.js allow the creation of data structures that can easily be displayed on a website. This and the ability to easily make calls to the backend make it a good fit for trckr.

3.6 CircleCI

CircleCI is a continuous integration and deployment platform, enabling teams to build software quickly and safely. Every time new code is committed, it runs a test suite to confirm that nothing broke and then immediately deploys this new version. Thus enabling features to be created quickly, without long and complicated deployments where certain bugs are discovered when the product should be going live. Using this technology enabled the trckr developers to develop features and immediately use them, motivating everybody to constantly improve the product in small steps and see and feel the results immediately.

4 Results

The architecture was kept fairly simple, while still providing a clear separation of concerns as illustrated in figure 1 using Django with PostgreSQL in the back-end and a front-end based around Vue.js.

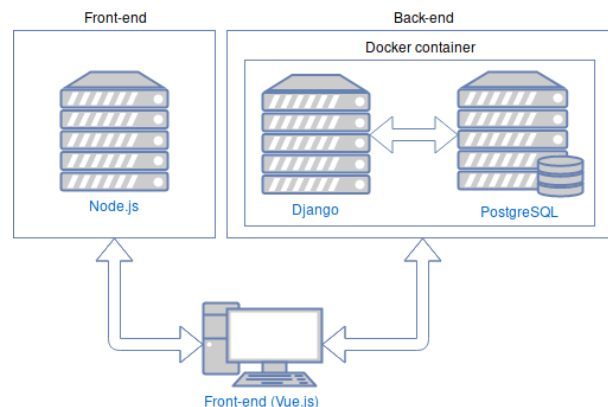


Figure 1: Architecture of trckr

4.1 API

The backend of the trckr application implements a RESTful API using the Django REST framework. The API provides basic CRUD operations for all the entities available in the database. There are five main endpoints to retrieve and save data on the server: authentication, user, projects, tasks and time entries. Except for the authentication and user endpoints, all endpoints need an authentication token to be accessed.

4.1.1 Endpoints

authentication allows users to retrieve an authentication token from the server to access the other parts of the API. Via this endpoint, one can also invalidate the token.

user is only used to create new user accounts.

projects allows users create, read, update and delete projects and display all the tasks associated with a project.

task is used to create, read and update tasks for a given project. There is also a way to list all relevant time entries for a task.

time entries are also used to create, read, update and delete operations

Each object of an entity has a unique ID. This ID can be used to retrieve information for that specific object by providing it in the URL when calling the server. This is necessary when updating an object via a POST request.

4.2 User Interface

are presented with the login screen as shown in figure 2. For users who have not yet registered an account, this can be done here by entering some basic information like username, password, email address and first and last name (Figure 3).

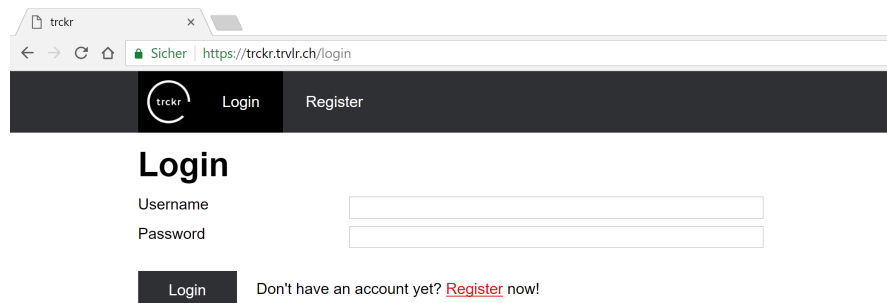


Figure 2: The trckr login page.

Once the registration process has been completed, the user can log in at the login page. Once the user is logged in, the navigation bar at the top of the interface will contain links to the dashboard, the projects page and the time entries page as well as a logout button. The dashboard displays different graphs to provide the user with a visual overview over their recent activities.

User registration

Username

Password

Email

First name

Last name

[Already have an account? Login!](#)

Figure 3: The trckr registration page.

This way the user can easily see, how much time was already tracked for each task and it enables the user to quickly proceed with their work, without having to search for a relevant task.

The projects page shows a table of all projects that the currently logged in user is a part of, this can be seen in figure 4. There is a search box above the projects table that allows a user to filter for a project or a group of projects containing the given keywords. To create a new project the user will have to navigate to the projects page and click on the "Create project" link which will open the project creation form. figure The form asks for the name of the project and optionally allows the user to enter a description.

Each project can be viewed in more detail when the project inside the table is clicked, as shown in figure 5. The project page contains the name, the description and a table of all the tasks in the selected project.

Projects

[Create project](#)

Filter projects

Project	Description
trckr	this is the trckr project
birthday party	*"CLASSIFIED" this is super secret

Figure 4: The projects page with the table containing the projects.

Clicking on a task will display its details like the name and description of the task. On the "Time Entries" page a user can create a time entry for a specific task of a project with the entry form shown in figure 6. The form asks the user to first choose a project and then a task of the project for which the time entry should be created.

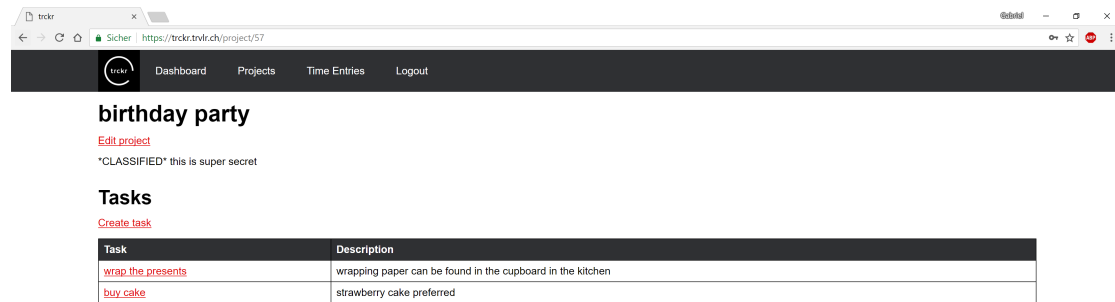


Figure 5: The project page with the table of all tasks.

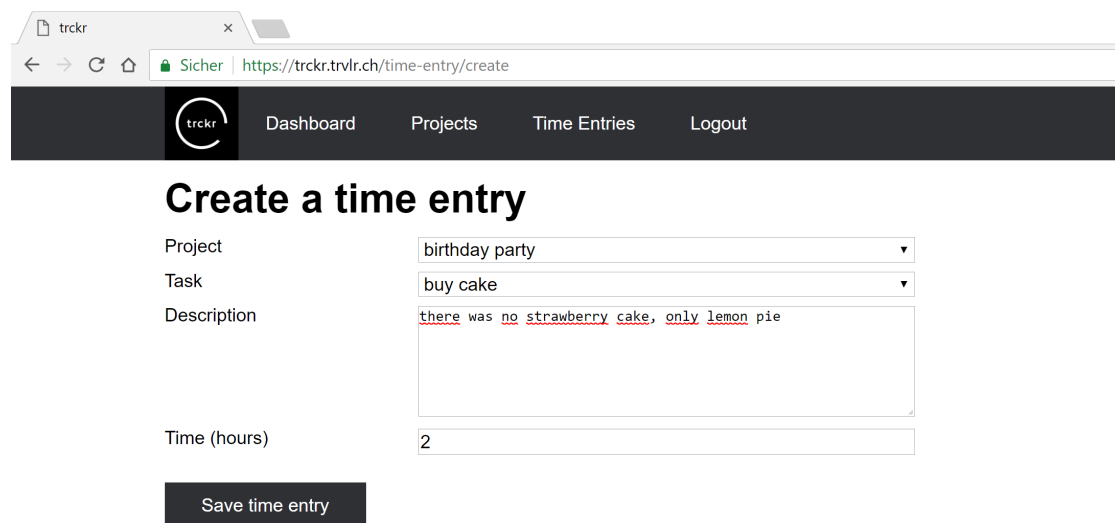


Figure 6: The form to add a time entry to a task.

All time entries will be displayed on the time entry page.

5 Outlook

By focusing on doing one thing and doing it right, it was possible to create a modular platform. That did not just allow a clean process during development so far, but also enables open source contributors to easily and properly add features in the future. In the first place, the application needs some real life usage in order to smooth the edges and fulfill the promise of being simple and fast to use. Nevertheless, some bigger feature sets were thought up that could not be implemented yet.

Project focused features can be improved to enable more specific project methods, maybe even interface with ticketing or project management tools. This enables the team leaders to gain more insight on the progress.

A desktop application could be developed to automatically track usage of certain applications, or integrations for IDEs to track time worked on a project. By reducing the manual intervention by the user, the tracking could become even more accurate. This might even uncover certain inefficiencies, say by detecting too much time waiting for tests or deployments.

6 Conclusion

The aim of project trckr was to create a tool, which helps people track their time on projects. With the help of Django in the backend and vuejs in the frontend, we developed a webapp with modern approaches and software developing methods. To integrate trckr into our productive environment, we used circleci, a very useful and automated integration tool. It helped us during the development phase target and solve bugs and other errors.

Getting started with vuejs was easy and fast for everyone, so developing single modules for the frontend was without big effort, beside testing.

References

- [1] Django. *Django The Web framework for perfectionists with deadlines*. 2018. URL: <https://www.djangoproject.com/> (visited on 10/05/2018).
- [2] PostgreSQL. *PostgreSQL The World's Most Advanced Open Source Relational Database*. 2018. URL: <https://www.postgresql.org/> (visited on 10/05/2018).
- [3] Docker. *Docker Build, Ship, and Run Any App, Anywhere*. 2018. URL: <https://www.docker.com/> (visited on 10/05/2018).
- [4] Node.js. *Node.js*. 2018. URL: <https://nodejs.org/> (visited on 10/05/2018).
- [5] Vue.js. *Vue.js The Progressive JavaScript Framework*. 2018. URL: <https://vuejs.org/> (visited on 10/05/2018).