Development of a modern frontend-backend user interface for Ricgraph (Research in context graph)

Rik D.T. Janssen, January 15, 2026

Introduction

In this project, the aim is to develop a modern frontend – backend user interface for Ricgraph. Ricgraph (www.ricgraph.eu), also known as Research in context graph, enables the exploration of researchers, teams, their results, collaborations, skills, projects, and the relations between these items.

Ricgraph can store many types of items into a single graph. These items can be obtained from various systems and from multiple organizations. Ricgraph facilitates reasoning about these items because it infers new relations between items, relations that are not present in any of the separate source systems. It is flexible and extensible, and can be adapted to new application areas.

In this project, we apply Ricgraph to the application area research information. Research information is about anything related to research: research results, the persons in a research team, their collaborations, their skills, projects in which they have participated, as well as the relations between these entities. Examples of research results are publications, data sets, and software.

Ricgraph is open source software and can be found on GitHub (https://github.com/UtrechtUniversity/ricgraph).

Current situation

Ricgraph Explorer (https://github.com/UtrechtUniversity/ricgraph/tree/main/ricgraph_explorer) is the web-based user interface to explore the information in Ricgraph. It is a Python application with some JavaScript that uses Flask to generate the HTML code to build a webpage. All of the HTML code is generated on the backend and then sent to the user's browser, that displays it. If the user wants to explore something based on this information, a request is sent to the backend, that generates a new webpage, that is sent to the browser, etc. Every time the backend sends a full webpage to the browser, even if only a small fraction of the content of that page has changed.

Envisioned situation

The envisioned situation for Ricgraph Explorer is a frontend – backend user interface, with minimal data exchange between frontend and backend. A user interacts with a webpage (the frontend), this request is sent to the backend, that only returns to the frontend the information that is new or that needs to be updated. Then, the frontend shows this to the user.

In this project, we will build a new frontend – backend user interface that has the same functionality as the current user interface of Ricgraph Explorer. The frontend will be new, the backend will be a modified version of the Python Flask code as it is in the current version of Ricgraph Explorer (to make it work with the new frontend). After this project has finished, there will be two user interfaces: the current user interface of Ricgraph Explorer, and the new frontend – backend to be build in this project. A user can choose between them. Of course they need to share code to make this possible.

In this project, React (https://react.dev) is used. For the template we use React-Bootstrap (https://react-bootstrap.netlify.app), unless the person working on the project proposes a different template/theme. The task is to create various reusable and nested components that are combined to user interfaces and web pages. To start, the components that exist in the current user interface of Ricgraph Explorer need to be created in React, basically rebuilding the user interface in React. Next, new components and web pages can be developed (that do not exist in Ricgraph Explorer).

Although React is a JavaScript library, various sources recommend using TypeScript (https://react.dev/learn/typescript) for React projects. TypeScript code is translated to JavaScript, which can then be executed by the user’s browser. Therefore, in this project, TypeScript will be used whenever possible.

Capabilities of the person running this project

- Experience in building user interfaces.
- Fluent in Python programming and Python Flask.
- Familiar with responsive web design (https://en.wikipedia.org/wiki/Responsive_web_design) and web accessibility (https://en.wikipedia.org/wiki/Web_accessibility).
- Knowledge of React, React templates, React themes.
- Capable of programming in TypeScript and JavaScript.