DIRACX-WEB:

The Next Evolution of the Dirac WebApp

D&RW 2023

October 18th 2023

Alexandre F. Boyer

alexandre.boyer@cern.ch

European Organization for Nuclear Research
Meyrin, Switzerland

Introduction

The current DIRAC Web Application

- Provides web interfaces to interact with DIRAC services.
- Has done the job so far...
- ...But will prevent us from moving forward in the next few years: under the hood, it is becoming unmaintainable.

DiracX is coming and represents a great opportunity to revisit the design and implementation of the web application.

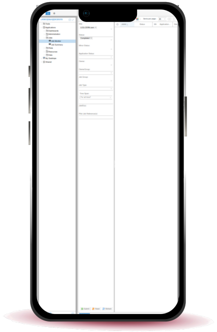
REQUIREMENTS

Limitations of the current DIRAC Web Application

What is wrong with the Web App?

- Highly custom: not based on a framework (not easy to modify, lack of support).
- Based on vendor lock-in libraries: components rely on ExtJS, which requires a custom compiler to work.
- Tightly coupled with DIRAC itself.

Technologies and libraries in web development are evolving rapidly, so it is worth taking a fresh look.



DiracX-Web requirements

Mostly follow DiracX requirements

- Technically: Welcoming to newcomers, responsive, stable and easy to deploy.
- Conceptually: Multi-VO, appealing, not overdesigned, user-centric and intuitive..

Based on the current limitations

- Should be based on a highly-used framework: it should be easy to get support from a community.
- Libraries should be open source and free: it should not be hard to escape from it if needed.

DiracX-Web current status

Foundations are being built

- The technical groundwork has begun.
- We want to provide a UI/UX that meets the needs of the DIRAC community: we need you (more details in a few slides).

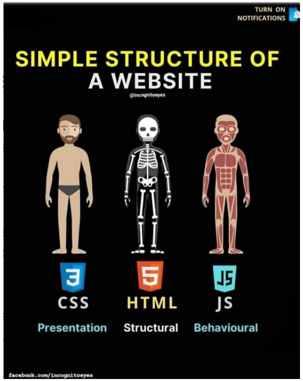
You can already have an overview of the source code:
https://github.com/DIRACGrid/diracx-web

TECHNOLOGIES INVOLVED

Introduction to front end development

Web browsers understand 3 languages

- HTML: provides a standardized means to specify web page content, such as text, photos, videos...
- CSS: used to determine web pages' visual appearance and layout.
- JavaScript: enables dynamic and interactive behavior on web pages (respond to user actions, perform calculations).



Building a web application without any library/framework on top of those would be time consuming and very challenging.

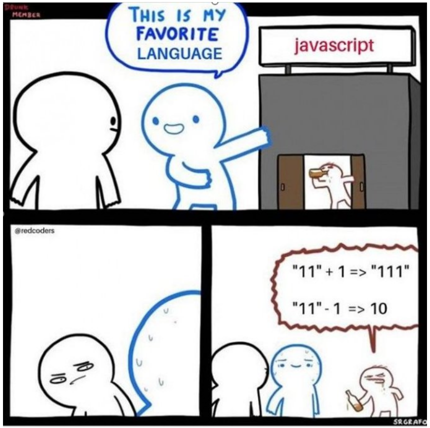
Using Typescript over Javascript

Typescript

- Superset of Javascript (code needs to be compiled).
- Appeared in 2012, developed by Microsoft.

Advantages

- Static Typing: more robust code, improved tooling (autocompletion), code documentation.
- Community and Ecosystem: strong community support and a growing ecosystem. Popular frameworks like Angular, Vue, and React support it.
- ESNext features: supports latest ECMAScript features (Javascript specifications).

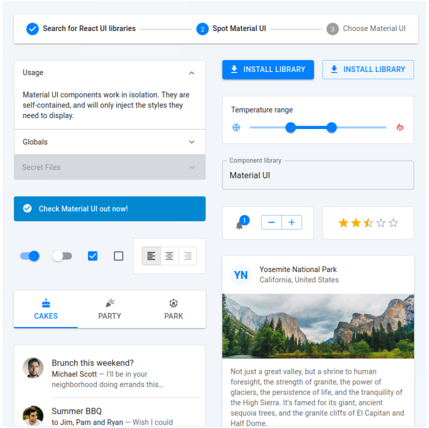


Using a component library instead of pure CSS

Designing nice and responsive UI components with HTML and CSS represents a massive amount of work (and great skills).

Material UI

- Library of components that can be directly imported and customized.
- Features an implementation of Google's Material Design system.
- Community and Ecosystem: strong community support and a growing ecosystem.



Material UI components work in isolation. They are self-contained, and will only inject the styles they need to display.

Brunch this weekend?
Michael Scott — I'll be in your neighborhood doing errands this…

Summer BBQ
to Jim, Pam and Ryan — Wish I could

Not just a great valley, but a shrine to human foresight, the strength of granite, the power of glaciers, the persistence of life, and the tranquility of the High Sierra. It's famed for its giant, ancient sequoia trees, and the granite cliffs of El Capitan and Half Dome.

Relying on a framework: React

Nowadays, most websites are built upon libraries and frameworks that allow teams to focus on components and business logic, and lean on battle-tested open-source solutions for routing, rendering, data fetching and more.

React

- The most famous Javascript/Typescript library nowadays.
- Created and used by Meta since 2011.
- Open source and large community and ecosystem: strong community support and a growing ecosystem.

Pros and Cons

+ Reusable UI components: can be combined to create complex components.
+ Virtual DOM: re-renders only the changed parts of the app.
- No convention: forces the development team to spend time on discussing some common development rules.
- Based on 3rd party libraries: no official libraries to handle common features such as routing, http requests...

Relying on a framework: NextJS

NextJS

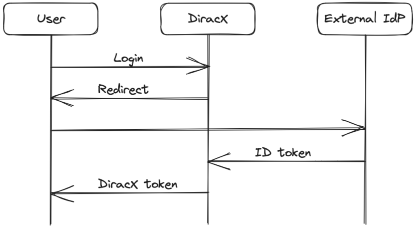
- Created & open sourced in 2016 by Vercel
- Community and Ecosystem: strong community support and a growing ecosystem.
- Out-of-the-box experience: structure, routing, rendering, data fetching, optimizations.
- Performant: applications built with it are fast.

Dealing with AuthN/AuthZ

Getting a token from an OIDC provider (through DiracX), extracting its information and making sure that only authorized users access critical parts of the web application is not trivial.

AXA oidc-client

- Library to manage authentication with the OpenID Connect (OIDC) and OAuth2 protocols.
- Works with any OIDC provider (very easy to setup).



Testing and keeping code consistent

Tests

It is essential to make sure that key user actions are working properly.

- Unit tests: JEST and React Testing Library.
- End to end testing: Cypress.

Code consistency

Avoid spending time on formatting rules and enforce a given level of code quality.

- ESLint: ensure code quality.
- Prettier: ensure code is properly formatted.
- Husky: pre-commit scripts.

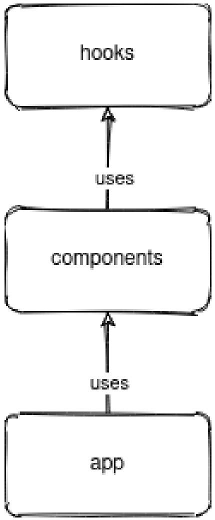
Technical state: Structure of the project

`/src`

- `hooks`: logic to build requests targeting a diracx installation. Investigating `autorest` from diracx.
- `components`: `ui` components (e.g. button, data table), `layout` (e.g. side bar, header) and applications (e.g. job monitor) are defined here.
- `app`: a way of organizing the components in a tree structure (e.g. `/src/app/dashboard` contains the content that will be displayed under https://<diracx>/dashboard).

`/test`

- `unit-tests`: test components.
- `integration-tests`: end-to-end tests (todo)



Technical state: conclusion

Technical foundations are (almost) here

- We rely on free, open source and widely used frameworks and libraries.
- We strive to provide a welcoming development environment for the contributors.

Now we need your feedback to develop a web UI/UX that would match your needs.

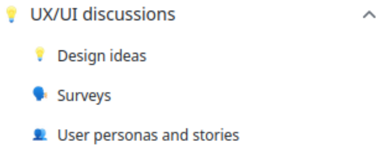
USER EXPERIENCE (UX) AND USER INTERFACE (UI)

Current design

The current DiracX web UI is poor in features: you can basically log in (single-VO) and monitor all your jobs.

How can you contribute?

- Want to give a general feedback on the DIRAC web application? Answer the survey.
- Want to request a feature? Create a user story to describe your need.
- Want to discuss about UX/UI design? Share your ideas.



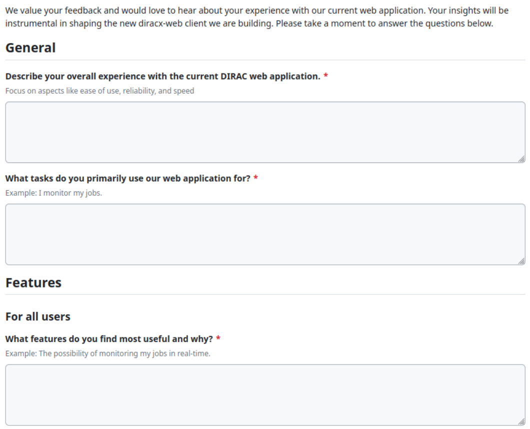
You can already submit your requests:
https://github.com/DIRACGrid/diracx-web/discussions

Survey

General feedback

Questions are related to the current DIRAC web application. Anyone can participate. We discuss about:

- General opinion of the web application.
- Features.
- UX/UI design and responsiveness.
- Extensions



We value your feedback and would love to hear about your experience with our current web application. Your insights will be instrumental in shaping the new diracx-web client we are building. Please take a moment to answer the questions below.

General

Describe your overall experience with the current DIRAC web application. *

Focus on aspects like ease of use, reliability, and speed

What tasks do you primarily use our web application for? *

Example: I monitor my jobs.

Features

For all users

What features do you find most useful and why? *

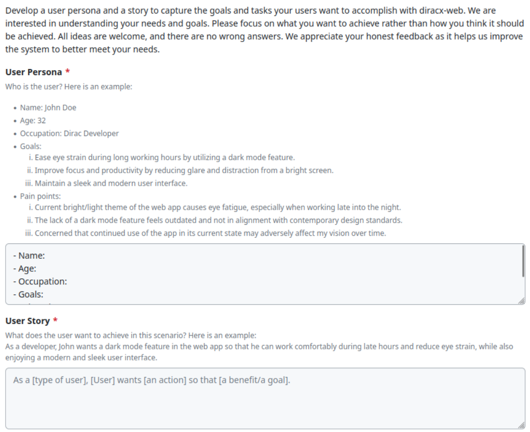
Example: The possibility of monitoring my jobs in real-time.

Feature Requests

User Personas and Stories

There is a clear way of requesting new features:

- Create a "persona": a semi-fictional representation of (some of) your users.
- Create a story: focus on what you want to achieve rather than how you think it should be achieved.
- An example is provided within the discussion template.



Develop a user persona and a story to capture the goals and tasks your users want to accomplish with diracx-web. We are interested in understanding your needs and goals. Please focus on what you want to achieve rather than how you think it should be achieved. All ideas are welcome, and there are no wrong answers. We appreciate your honest feedback as it helps us improve the system to better meet your needs.

User Persona *

Who is the user? Here is an example:

- Name: John Doe
- Age: 32
- Occupation: Dirac Developer
- Goals:
  i. Ease eye strain during long working hours by utilizing a dark mode feature.
  ii. Improve focus and productivity by reducing glare and distraction from a bright screen.
  iii. Maintain a sleek and modern user interface.
- Pain points:
  i. Current bright/light theme of the web app causes eye fatigue, especially when working late into the night.
  ii. The lack of a dark mode feature feels outdated and not in alignment with contemporary design standards.
  iii. Concerned that continued use of the app in its current state may adversely affect my vision over time.

```
- Name:
- Age:
- Occupation:
- Goals:
```

User Story *

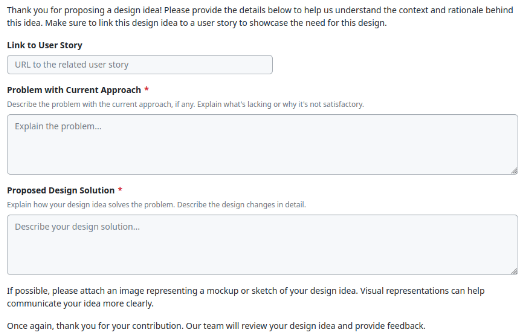
What does the user want to achieve in this scenario? Here is an example:
As a developer, John wants a dark mode feature in the web app so that he can work comfortably during late hours and reduce eye strain, while also enjoying a modern and sleek user interface.

```
As a [type of user], [User] wants [an action] so that [a benefit/a goal].
```

Design ideas

Based on an existing story, you might want to propose design choices

- You need to link your design idea to an existing user story.
- You need to describe the problem you have with the current approach...
- ...And explain the reasons for your choices. Don't hesitate to include drawings that reflect your ideas (tools like Ninja Mock can be helpful).



Thank you for proposing a design idea! Please provide the details below to help us understand the context and rationale behind this idea. Make sure to link this design idea to a user story to showcase the need for this design.

Link to User Story

URL to the related user story

Problem with Current Approach *

Describe the problem with the current approach, if any. Explain what's lacking or why it's not satisfactory.

Explain the problem...

Proposed Design Solution *

Explain how your design idea solves the problem. Describe the design changes in detail.

Describe your design solution...

If possible, please attach an image representing a mockup or sketch of your design idea. Visual representations can help communicate your idea more clearly.

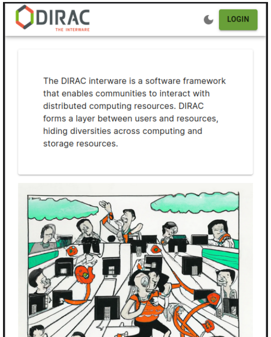
Once again, thank you for your contribution. Our team will review your design idea and provide feedback.

CONCLUSION

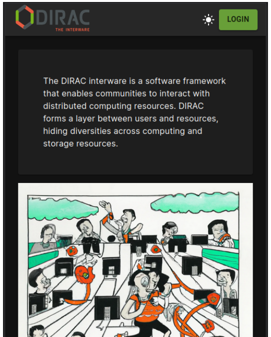
Want to get an overview?

DIY

- Clone the DiracX chart: `git clone https://github.com/DIRACGrid/diracx-charts`
- Run the demo: `diracx-charts/run_demo.sh` (not now!)
- Copy paste the obtained URL into your web browser.



The DIRAC interware is a software framework that enables communities to interact with distributed computing resources. DIRAC forms a layer between users and resources, hiding diversities across computing and storage resources.



The DIRAC interware is a software framework that enables communities to interact with distributed computing resources. DIRAC forms a layer between users and resources, hiding diversities across computing and storage resources.

Conclusion

Main contribution

- The foundations of the next DIRAC web application: DiracX-Web.
- An overview of the project technicalities.
- Explanations on how to contribute to the web UX/UI.

What's next?

- We need your feedback! We are waiting for your feature requests and design ideas.

Thank you for your attention

Questions? Comments?