refinedev / **refine** Public

Build your React-based CRUD applications, without constraints.

refine.dev

MIT license

8.6k stars **760** forks

Star | Notifications

Code | Issues 23 | Pull requests 7 | Discussions | Actions | Projects 2 | Security | In

next

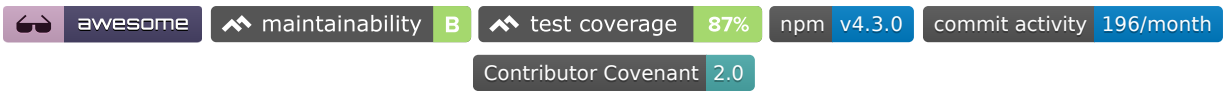
omeraplak Merge branch 'next' of github.com:refinedev/refine into next ... 1 hour ago **4,515**

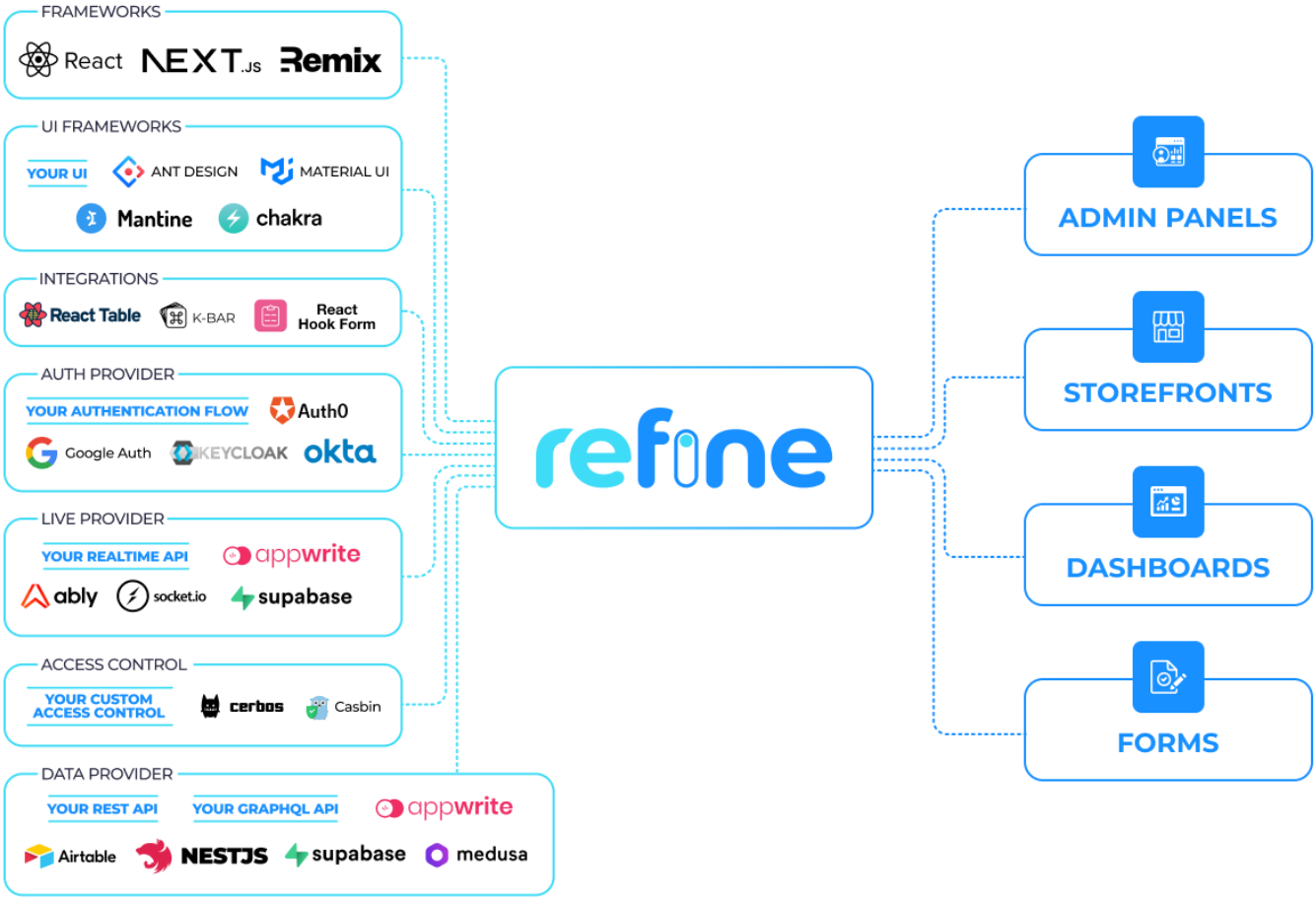
View code

README.md

Home Page | Discord | Examples | Blog | Documentation | Roadmap

Build your React-based CRUD applications, without constraints.
An open source, headless web application framework developed with flexibility in mind.





What is refine?

refine is a React-based framework for the rapid ✨ development of web applications. It eliminates repetitive tasks demanded by **CRUD** operations and provides industry standard solutions for critical parts like **authentication**, **access control**, **routing**, **networking**, **state management**, and **i18n**.

refine is *headless by design*, thereby offering unlimited styling and customization options.

What do you mean by "headless" ?

Instead of being a limited set of pre-styled components, **refine** is a collection of helper `hooks`, `components`, and `providers`. They are all decoupled from *UI components* and *business logic*, so that they never keep you from customizing your *UI* or coding your own flow.

refine seamlessly works with any **custom design** or **UI framework** that you favor. For convenience, it ships with ready-made integrations for Ant Design System, Material UI, Mantine, and Chakra UI.

Use cases

refine shines on *data-intensive*⚡ applications like **admin panels**, **dashboards** and **internal tools**. Thanks to the built-in **SSR support**, **refine** can also power *customer-facing* applications like **storefronts**.

You can take a look at some live examples that can be built using **refine** from scratch:

👉 Refer to most popular real use case examples

👉 More **refine** powered different usage scenarios can be found here

Key Features

⚙️ Zero-config, **one-minute setup** with a **single CLI command**

🔌 Connectors for **15+ backend services** including REST API, GraphQL, NestJs CRUD, Airtable, Strapi, Strapi v4, Strapi GraphQL, Supabase, Hasura, Appwrite, Firebase, and Directus.

🌐 **SSR support** with **Next.js** or **Remix**

🔍 Auto-generated **CRUD** UIs from **your API data structure**

⚛️ Perfect **state management** & **mutations** with **React Query**

🔀 **Advanced routing** with any router library of your choice

🔐 Providers for seamless **authentication** and **access control** flows

⚡ Out-of-the-box support for **live / real-time applications**

📄 Easy **audit logs** & **document versioning**

💬 Support for any **i18n** framework

💪 Future-proof, **robust architecture**

⏳ Built-in CLI with time-saving features

✔️ Full **test coverage**

Quick Start

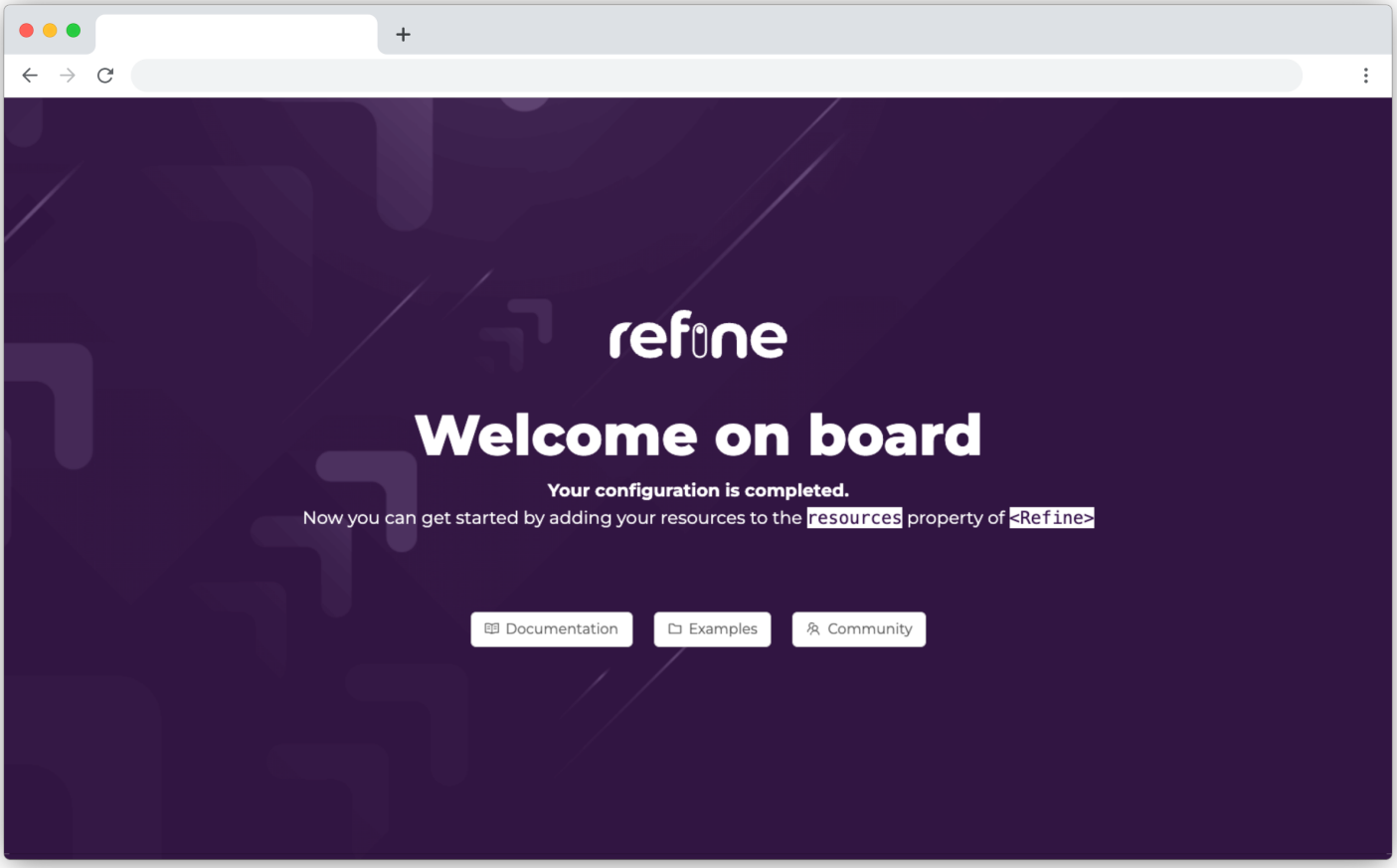
The fastest way to get started with **refine** is by using the `create refine-app` project starter tool. Run the following command to create a new **refine** project configured with Ant Design System as the default UI framework:

```
npm create refine-app@latest -- -o refine-antd
```

Once the setup is complete, navigate to the project folder and start your project with:

```
npm run dev
```

Your **refine** application will be accessible at http://localhost:3000:



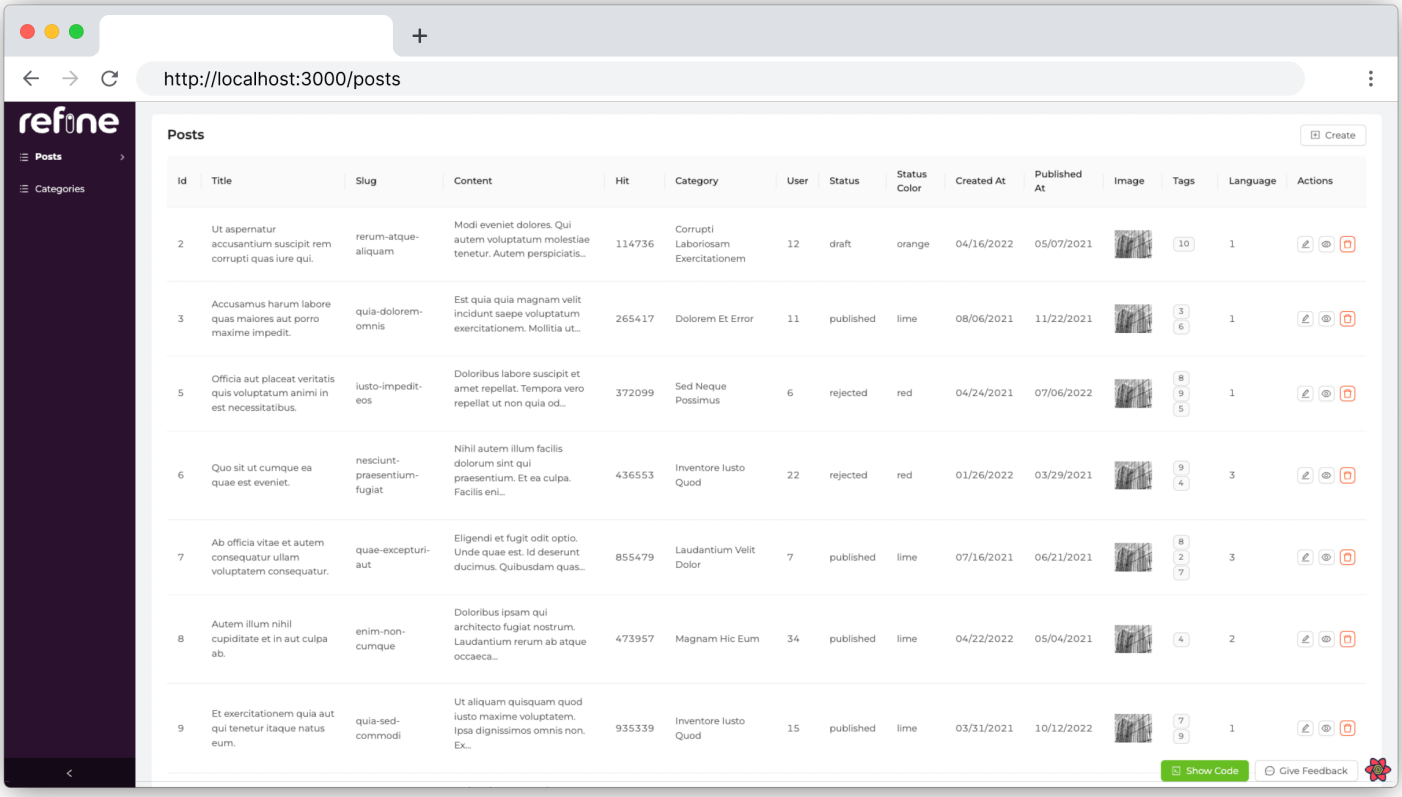
Let's consume a public `fake REST API` and add two resources (*posts*, *categories*) to our project. Replace the contents of `src/App.tsx` with the following code:

```
import { Refine } from "@refinedev/core";
import { Layout, notificationProvider, ErrorComponent } from "@refinedev/antd";
import routerProvider, { NavigateToResource } from "@refinedev/react-router-v6";
import dataProvider from "@refinedev/simple-rest";

import { BrowserRouter, Routes, Route, Outlet } from "react-router-dom";

import { AntdInferencer } from "@refinedev/inferencer/antd";

import "@refinedev/antd/dist/reset.css";

const App: React.FC = () => {
    return (
        <BrowserRouter>
            <Refine
                routerProvider={routerProvider}
                dataProvider={dataProvider("https://api.fake-rest.refine.dev")}
                notificationProvider={notificationProvider}
                resources={[
                    {
                        name: 'posts',
                        list: "/posts",
                        show: "/posts/show/:id",
                        create: "/posts/create",
                        edit: "/posts/edit/:id",
                        meta: { canDelete: true },
```

```
                    },
                    {
                        name: 'categories',
                        list: "/categories",
                        show: "/categories/show/:id",
                    }
                ]}
            >
                <Routes>
                    <Route
                        element={(
                            <Layout>
                                <Outlet />
                            </Layout>
                        )}
                    >
                        <Route index element={<NavigateToResource />} />
                        <Route path="posts">
                            <Route index element={<AntdInferencer />} />
                            <Route path="show/:id" element={<AntdInferencer />} />
                            <Route path="create" element={<AntdInferencer />} />
                            <Route path="edit/:id" element={<AntdInferencer />} />
                        </Route>
                        <Route path="categories">
                            <Route index element={<AntdInferencer />} />
                            <Route path="show/:id" element={<AntdInferencer />} />
                        </Route>
                        <Route path="*" element={<ErrorComponent />} />
                    </Route>
                </Routes>
            </Refine>
        </BrowserRouter>
    );
};

export default App;
```

🚀 Thanks to **refine Inferencer package**, it guesses the configuration to use for the `list`, `show`, `create`, and `edit` pages based on the data fetched from the API and generates the pages automatically.

Now, you should see the output as a table populated with `post` & `category` data:



You can get the auto-generated pages codes by clicking the `Show Code` button on each page. Afterward, simply pass the pages to the `resources` array by replacing with the Inferencer components.

Next Steps

👉 Jump to Tutorial to continue your work and turn the example into a full-blown CRUD application.

👉 Visit Learn the Basics Page to get informed about the fundamental concepts.

👉 Read more on Advanced Tutorials for different usage scenarios.

👉 See the real-life Finefoods Demo project.

👉 Play with interactive Examples

Roadmap

You can find refine's Public Roadmap here!

Stargazers

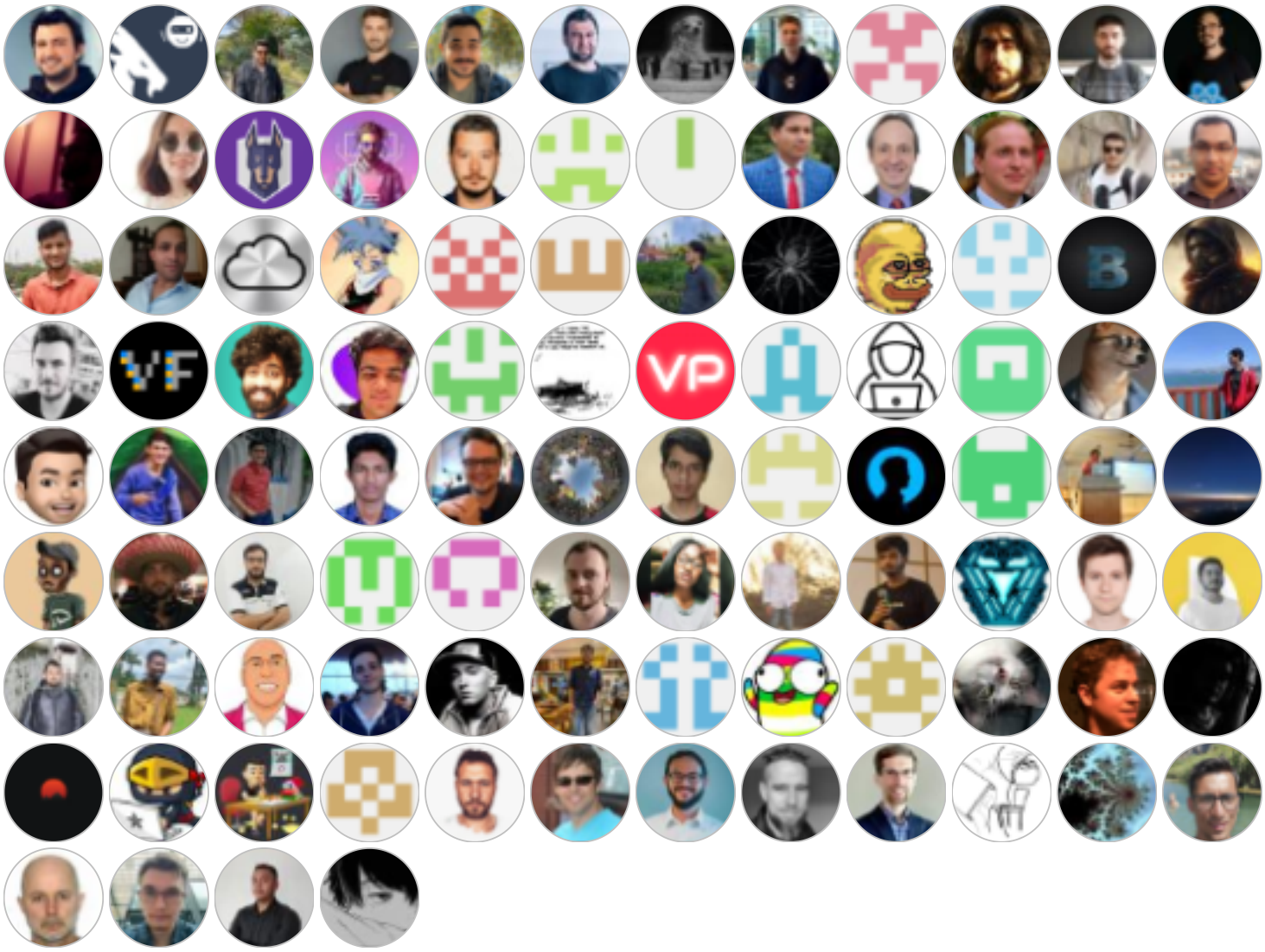
@whitejsx, **@jtweis**, **@jesusbello**, and **8,631 others** starred this repository.

Contribution

👉 Refer to contribution docs for more information

If you have any doubts related to the project or want to discuss something, then join our Discord Server.

Our ❤️ Contributors



License

Releases 803

@refinedev/ui-tests@1.1.2 Latest
yesterday

+ 802 releases

Sponsor this project

Sponsor

Learn more about GitHub Sponsors

Used by 1.3k

+ 1,247

Contributors 135

+ 124 contributors

Languages

- **TypeScript** 97.1%
- **JavaScript** 2.7%
- **CSS** 0.2%
- **Handlebars** 0.0%
- **Smarty** 0.0%
- **Dockerfile** 0.0%