

Fullstack Development

Authentication / Authorization

Part 4: SPA

Setup

- `git clone -b main https://github.com/fullstack-67/auth-spa.git auth-spa`
- Backend
 - `cd backend`
 - Fill in `.env`
 - `pnpm install`
 - `npm run db:reset`
 - `npm run dev`

Setup

- Frontend
 - `cd frontend`
 - `pnpm install`
 - `npm run dev`

Backend

- Routes now return json / redirect header instead of html.
- Created `GET /me` route for clients to check their `auth` states.
- Apart from that, there is very minimal change, surprisingly.

FileEditSelectionViewGoRunTerminalHelp

auth-spa

index.mpa.ts ↔ index.ts X

backend > src > index.ts > ...

22 hidden lines

23 app.use(passportIns.session());

24

25 // * Endpoints

26 app.get("/", async (req, res, next) => {

27 const sessions = await formatSession(req);

28 res.render("pages/index", {

29 title: "Home",

30 user: req.user,

31 sessions: sessions,

32 });

33 });

34

35 app.get("/signup", function (req, res) {

34 hidden lines

70 app.post("/login", passportIns.authenticate("local"), function (req, res) {

71 debug("@login handler");

72 setSessionInfoAfterLogin(req, "CREDENTIAL");

73 res.setHeader("HX-Redirect", "/");

74 res.send("<div></div>");

75 });

76

77 app.get("/login/oauth/github", passportIns.authenticate("github"));

20 hidden lines | app.get("/callback/google") callback

98 }

99);

100

101 app.post("/logout", function (req, res, next) {

102 // req.logout will not delete the session in db. It will generate new one for the

103 // When the user login again, it will generate new session with the user id.

104 req.logout(function (err) {

5 hidden lines | req.session.destroy() callback

110 if (err) {

111 return next(err);

112 }

113 res.setHeader("HX-Redirect", "/");

114 res.send("<div></div>");

115 });

116 });

117 });

13 hidden lines

22 hidden lines

23 app.use(passportIns.session());

24

25 // * Endpoints

26+ app.get("/me", async (req, res, next) => {

27 const sessions = await formatSession(req);

28+ const user = req?.user ?? null;

29+ res.json({ sessions, user });

30 });

31

32 app.get("/signup", function (req, res) {

34 hidden lines

67 app.post("/login", passportIns.authenticate("local"), function (req, res) {

68 debug("@login handler");

69 setSessionInfoAfterLogin(req, "CREDENTIAL");

70+ res.status(200).json("Login Successful");

71 });

72

73 app.get("/login/oauth/github", passportIns.authenticate("github"));

20 hidden lines | app.get("/callback/google") callback

94 }

95);

96

97+ app.get("/logout", function (req, res, next) {

98 // req.logout will not delete the session in db. It will generate new one for the alr

99 // When the user login again, it will generate new session with the user id.

100 req.logout(function (err) {

5 hidden lines | req.session.destroy() callback

106 if (err) {

107 return next(err);

108 }

109+ res.redirect("/");

110 });

111 });

112 });

13 hidden lines

Frontend

- Created (client-side) routing.
- Created logic to query/update `auth`'s state.
- Modified proxy server to take care of `Callback URL` (bypass client-routing).
- Created signup form/login and other UI.

Highlighted packages

```
{  
  "@tanstack/react-query": "^5.52.1",  
  "react-router-dom": "^6.26.1"  
}
```

Client-side routing

src/App.tsx

```
import { createBrowserRouter, RouterProvider } from "react-router-dom";
const router = createBrowserRouter([
  {
    path: "/",
    element: <Layout />,
    children: [
      {
        path: "/",
        element: <Home />,
      },
    ],
  },
]);
```

Client-side routing

src/App.tsx

```
function App() {  
  return (  
    // ...  
    <RouterProvider router={router} />  
    // ...  
  );  
}  
  
export default App;
```

Getting auth state

src/hooks/useAuth.ts

```
import { useQuery } from "@tanstack/react-query";
// ...
function getMe() {
  return axios.get<AuthData>("/api/me");
}

function useAuth() {
  // Queries
  const { data, error, refetch } = useQuery({
    queryFn: getMe,
    // Other options
  });
  return { user: data?.user, sessions: data?.sessions, error, refetch };
}
```

Getting auth state

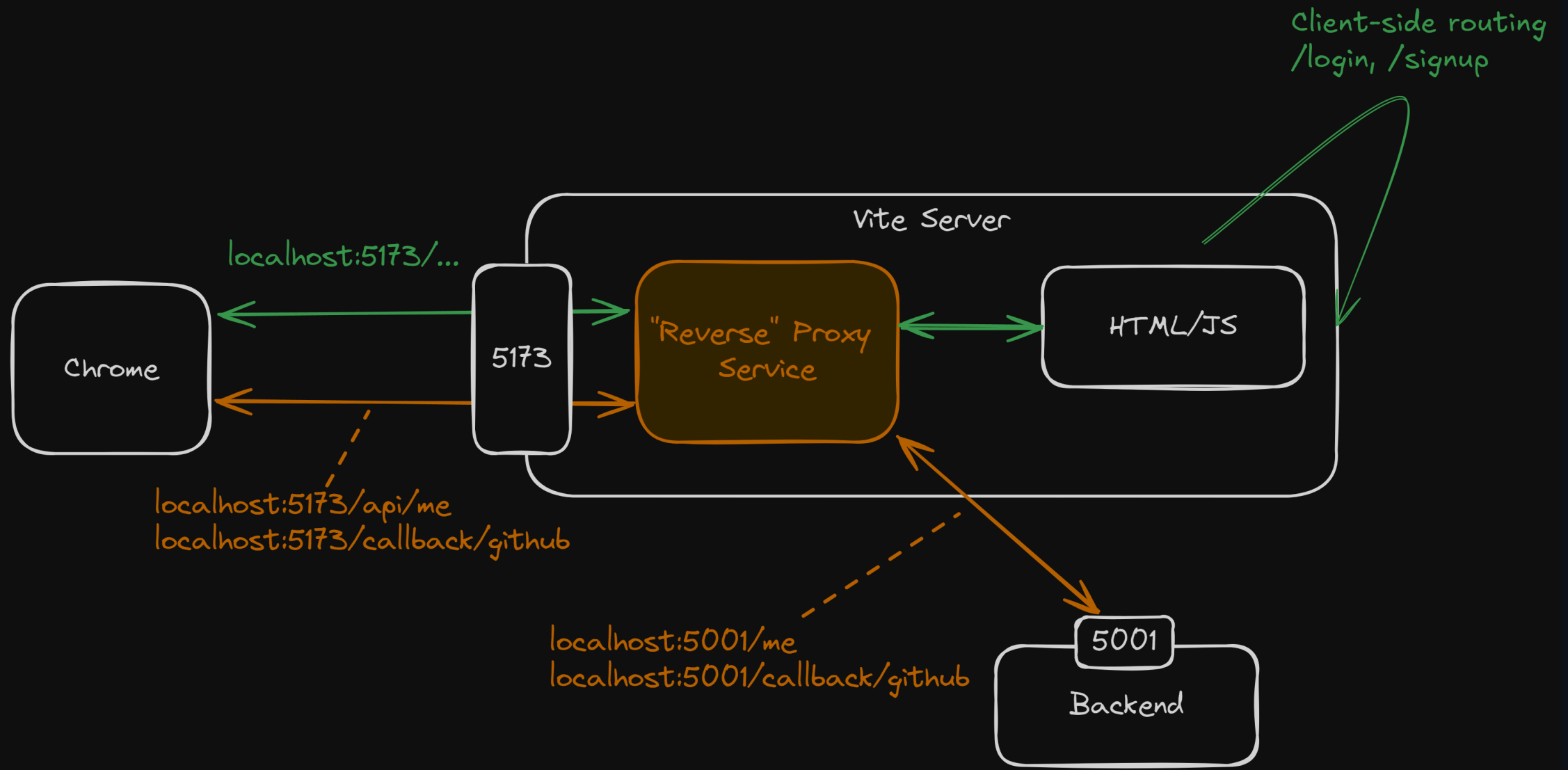
src/components/Nav.tsx

```
const Nav: FC = () => {  
  const { user } = useAuth(); ➡➡➡  
  // ...  
  return (  
    <nav>  
      // ...  
    </nav>  
  );  
};  
  
export default Nav;
```

Handle Callback URL

vite.config.ts

```
export default defineConfig({
  plugins: [react()],
  server: {
    proxy: {
      "/api": {
        // ...
      },
      "/callback": { ➡➡➡
        target: "http://localhost:5002",
      },
    },
  },
});
```



With proxy server

- The cookie will automatically sent to backend for all requests because it is the "same site".

If you don't have proxy server.

- Cookies are still automatically sent given the same host but different port.
- Need to set `withCredentials` to `true` on any AJAX request (via `fetch` or `axios` APIs)
 - Note that `httpOnly` cookie are sent by this technique.
- Need to allow CORS in the backend.

Auth.js

Setup

- `git clone -b authjs https://github.com/fullstack-67/auth-mpa-v2.git auth-authjs`
- `pnpm i`
- `npm run db:reset`
- `npm run dev`

Note

- DB schema
- Available auth methods
- Check cookie after login.