Fullstack Development

Authentication / Authorization

Part 4: SPA

Setup

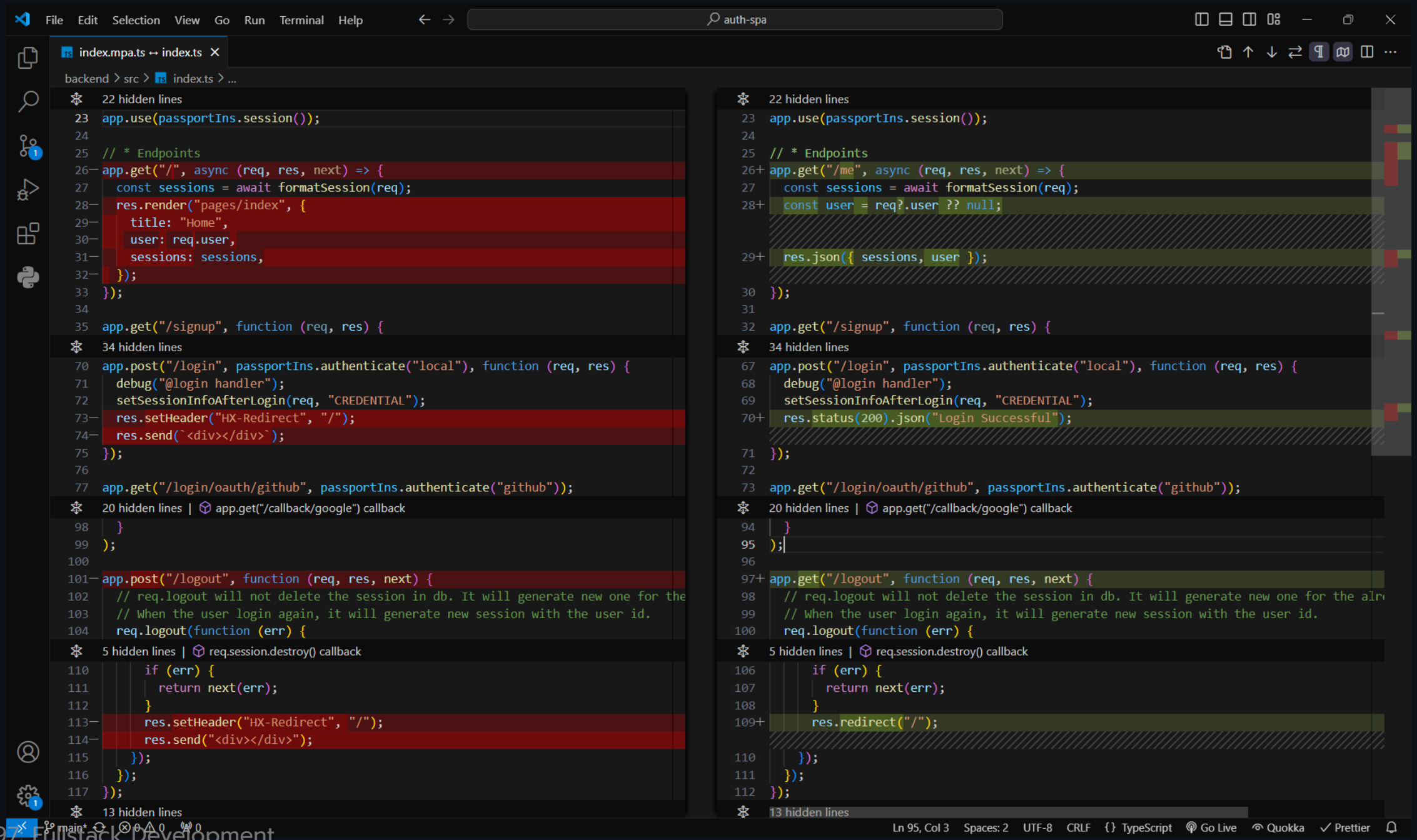
- `git clone -b main https://github.com/fullstack-67/auth-spa.git auth-spa`
- Backend
  - `cd backend`
  - Fill in `.env`
  - `pnpm install`
  - `npm run db:reset`
  - `npm run dev`

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 mininal change, surprisingly.


File Edit Selection View Go Run Terminal Help
auth-spa
index.mpa.ts ↔ index.ts
backend > src > index.ts > ...
22 hidden lines
app.use(passportIns.session());
// * Endpoints
app.get("/", async (req, res, next) => {
const sessions = await formatSession(req);
res.render("pages/index", {
title: "Home",
user: req.user,
sessions: sessions,
});
});
app.get("/signup", function (req, res) {
34 hidden lines
app.post("/login", passportIns.authenticate("local"), function (req, res) {
debug("@login handler");
setSessionInfoAfterLogin(req, "CREDENTIAL");
res.setHeader("HX-Redirect", "/");
res.send(`<div></div>`);
});
app.get("/login/oauth/github", passportIns.authenticate("github"));
20 hidden lines | app.get("/callback/google") callback
}
);
app.post("/logout", function (req, res, next) {
// req.logout will not delete the session in db. It will generate new one for the
// When the user login again, it will generate new session with the user id.
req.logout(function (err) {
5 hidden lines | req.session.destroy() callback
if (err) {
return next(err);
}
res.setHeader("HX-Redirect", "/");
res.send(" ");
});
});
});
13 hidden lines
app.use(passportIns.session());
// * Endpoints
app.get("/me", async (req, res, next) => {
const sessions = await formatSession(req);
const user = req?.user ?? null;
res.json({ sessions, user });
});
app.get("/signup", function (req, res) {
34 hidden lines
app.post("/login", passportIns.authenticate("local"), function (req, res) {
debug("@login handler");
setSessionInfoAfterLogin(req, "CREDENTIAL");
res.status(200).json("Login Successful");
});
app.get("/login/oauth/github", passportIns.authenticate("github"));
20 hidden lines | app.get("/callback/google") callback
}
);
app.get("/logout", function (req, res, next) {
// req.logout will not delete the session in db. It will generate new one for the alr
// When the user login again, it will generate new session with the user id.
req.logout(function (err) {
5 hidden lines | req.session.destroy() callback
if (err) {
return next(err);
}
res.redirect("/");
});
});
});
13 hidden lines
main*
Ln 95, Col 3
Spaces: 2
UTF-8
CRLF
TypeScript
Go Live
Quokka
Prettier

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

```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

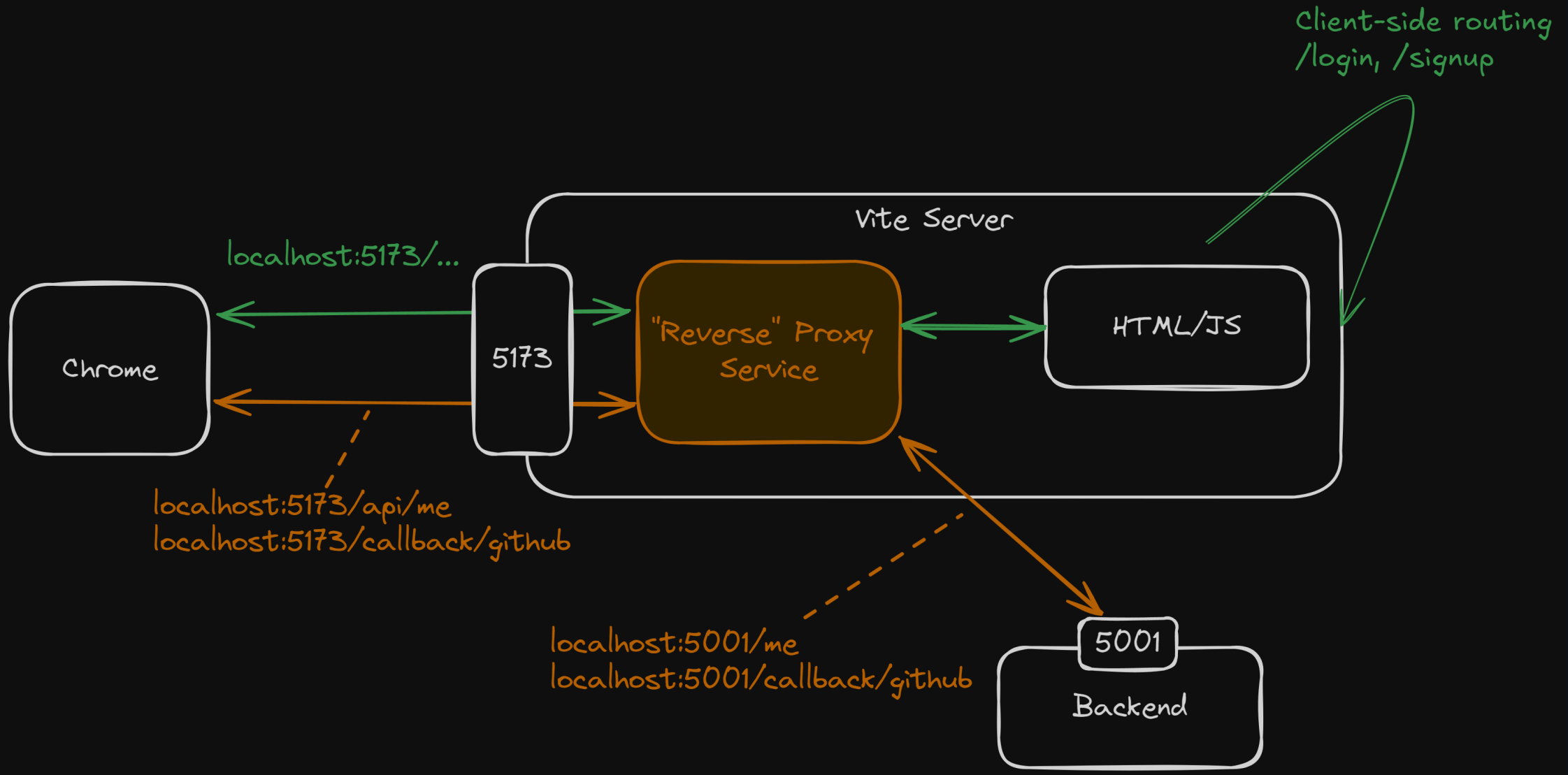
```tsx
const Nav: FC = () => {
  const { user } = useAuth(); 👀👀👀
  // ...
  return (
    <nav>
      // ...
    </nav>
  );
};

export default Nav;
```

Handle `Callback URL`

vite.config.ts

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


Client-side routing
/login, /signup
Vite Server
localhost:5173/...
Chrome
5173
"Reverse" Proxy
Service
HTML/JS
localhost:5173/api/me
localhost:5173/callback/github
localhost:5001/me
localhost:5001/callback/github
5001
Backend

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.