Full Stack Implementation

Implementing SvelteKit full stack apps

Donation-svelte-08-full

File tree (left panel):
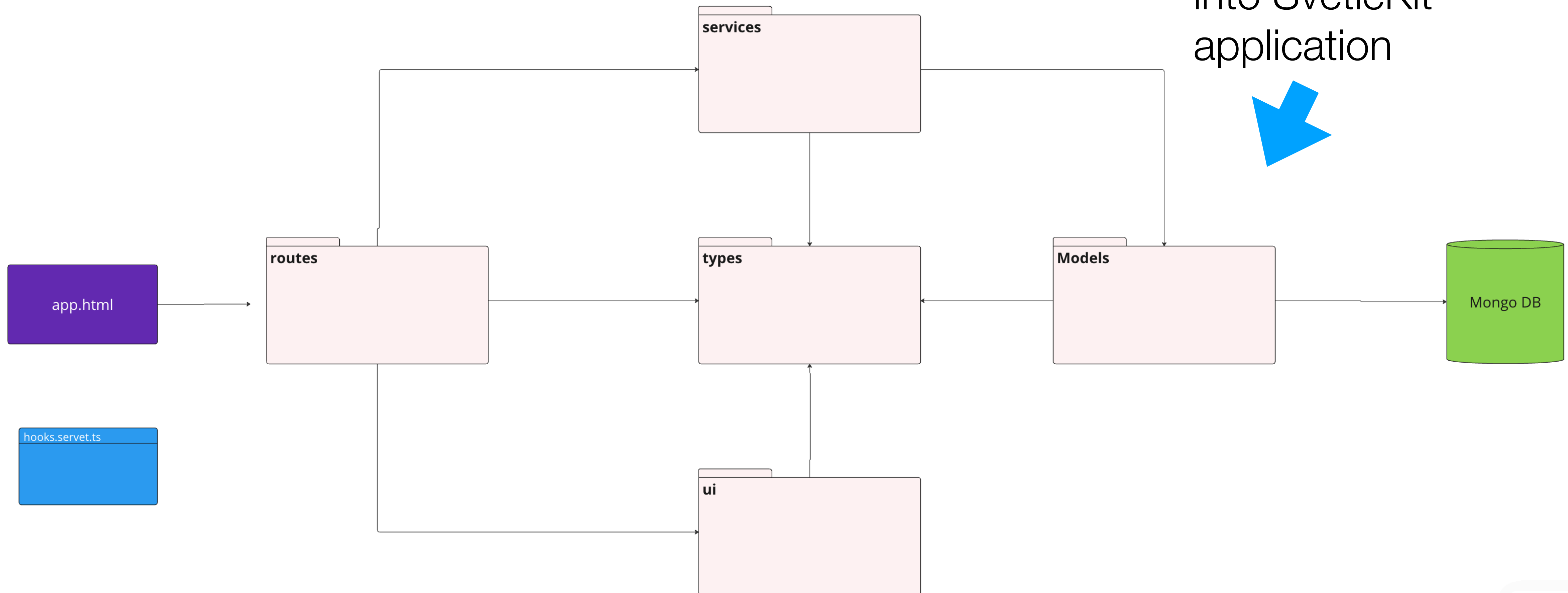
- routes
  - charts
    - TS +page.server.ts
    - +page.svelte
  - donate
    - TS +page.server.ts
    - +page.svelte
    - DonateForm.svelte
  - login
    - TS +page.server.ts
    - +page.svelte
    - LoginForm.svelte
  - logout
    - TS +page.server.ts
    - +page.svelte
  - maps
    - TS +page.server.ts
    - +page.svelte
  - report
    - TS +page.server.ts
    - +page.svelte
  - signup
    - +page.svelte
    - SignupForm.svelte
  - TS +layout.server.ts
  - +layout.svelte
  - +page.svelte
  - TS app.d.ts
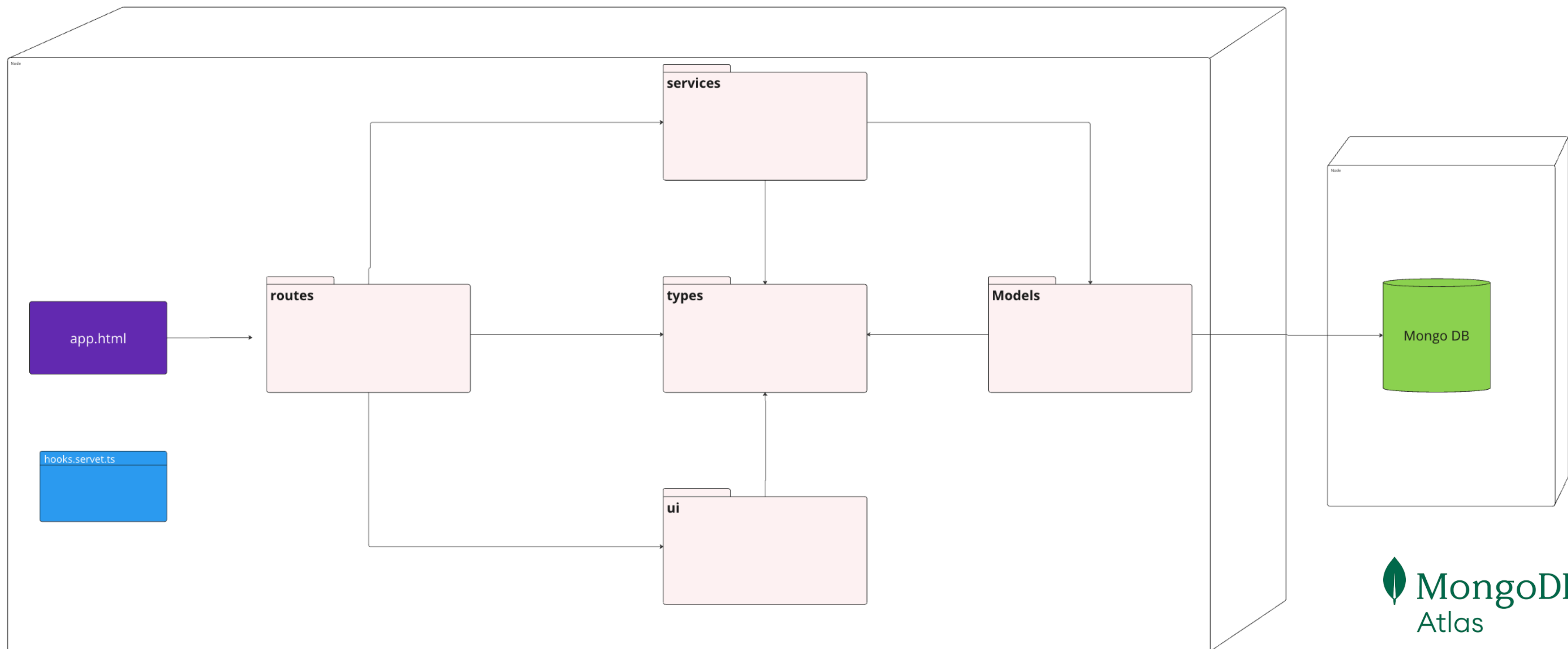 - app.html
  - TS hooks.server.ts
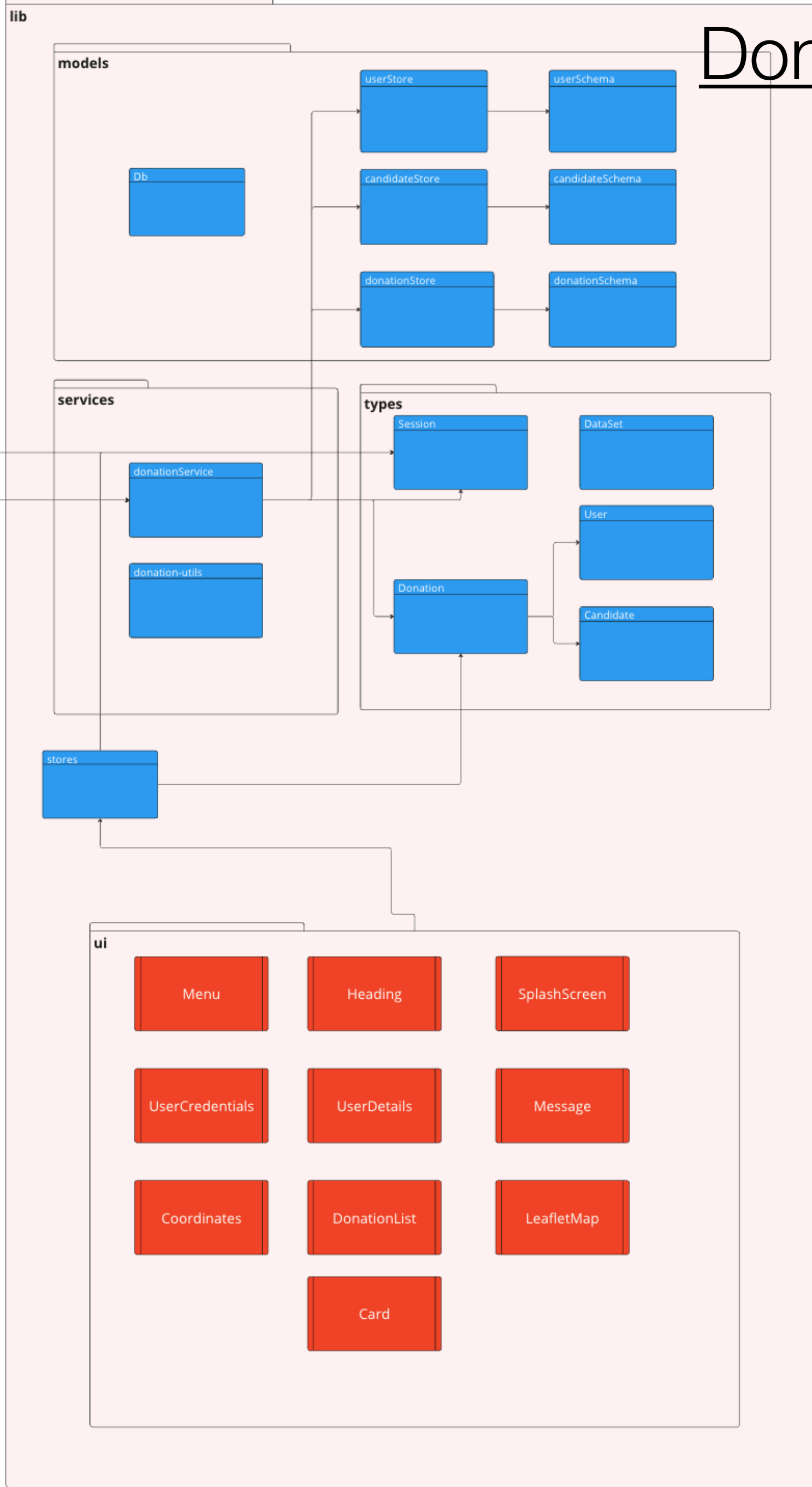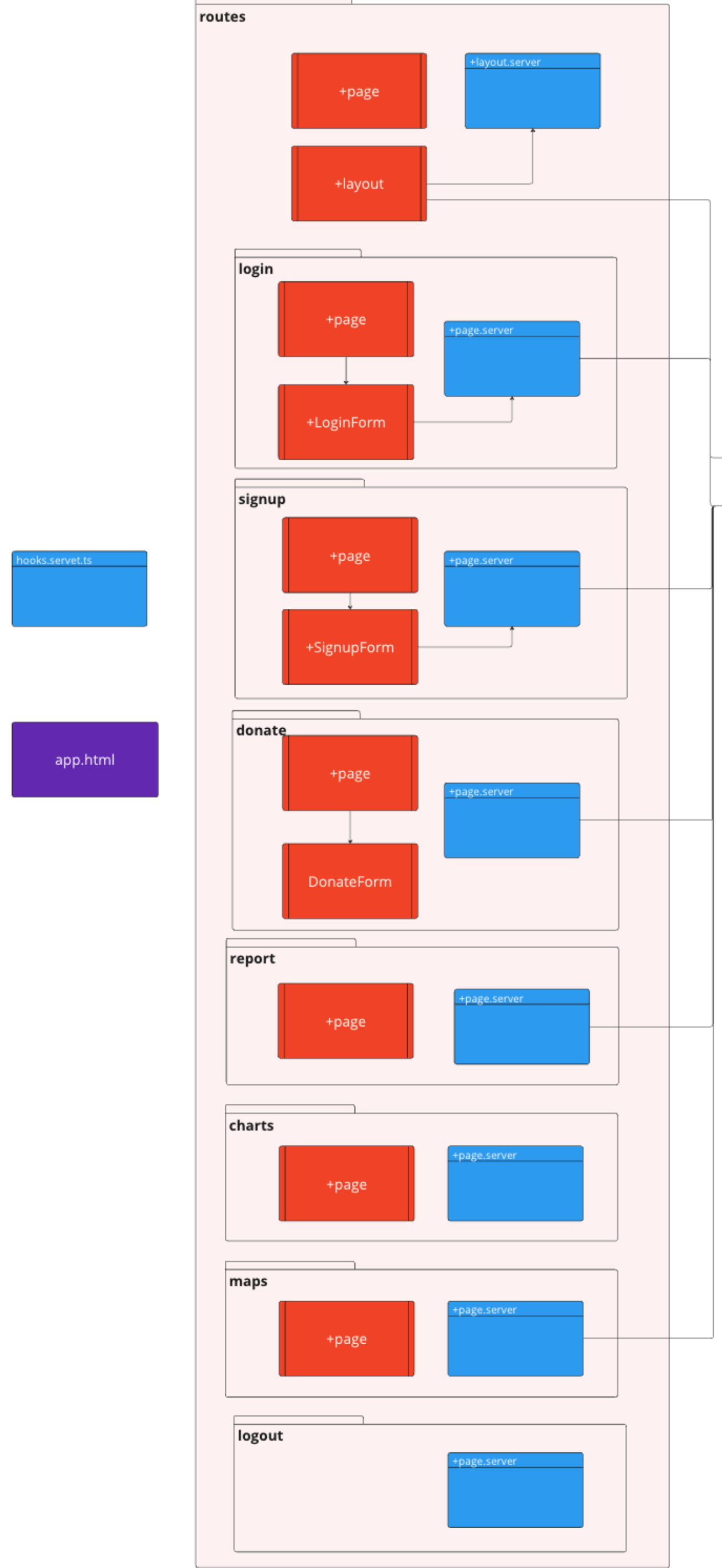
File tree (right panel):

- src
  - lib
    - models
      - mongo
        - TS candidate-store.ts
        - TS candidate.ts
        - TS donation-store.ts
        - TS donation.ts
        - TS user-store.ts
        - TS user.ts
      - TS db.ts
    - services
      - TS donation-service.ts
      - TS donation-utils.ts
    - types
      - TS donation-types.ts
    - ui
      - Card.svelte
      - Coordinates.svelte
      - DonationList.svelte
      - Heading.svelte
      - LeafletMap.svelte
      - Menu.svelte
      - Message.svelte
      - SplashScreen.svelte
      - UserCredentials.svelte
      - UserDetails.svelte
    - TS stores.ts

Diagram — routes:
- +page
- +layout.server
- +layout
- login: +page, +LoginForm, +page.server
- signup: +page, +SignupForm, +page.server
- donate: +page, DonateForm, +page.server
- report: +page, +page.server
- charts: +page, +page.server
- maps: +page, +page.server
- logout: +page.server
- hooks.servet.ts
- app.html

Diagram — lib:
- models: Db, userStore, userSchema, candidateStore, candidateSchema, donationStore, donationSchema
- services: donationService, donation-utils
- types: Session, DataSet, User, Donation, Candidate
- stores
- ui: Menu, Heading, SplashScreen, UserCredentials, UserDetails, Message, Coordinates, DonationList, LeafletMap, Card

Donation Full Stack Architecture

- Incorporate Models package from donation-api directly into SvetleKit application

services

routes

**types**

Models

app.html

Mongo DB

hooks.servet.ts

ui

Donation Full Stack Deployment

Migrate Mongoose Models

```
v 📁 src
  v 📁 api
      TS candidates-api.ts
      TS donations-api.ts
      TS jwt-utils.ts
      TS users-api.ts
  v 📁 controllers
      TS accounts-controller.ts
      TS donations-controller.ts
  v 📁 models
    v 📁 mongo
        TS candidate-store.ts
        TS candidate.ts
        TS connect.ts
        TS donation-store.ts
        TS donation.ts
        TS seed-data.ts
        TS user-store.ts
        TS user.ts
      TS db.ts
  v 📁 types
      TS donation-types.ts
```

- From Donation-hapi app

- … to SvelteKit app

```
npm install mongoose
```

```
v 📁 src
  v 📁 lib
    v 📁 models
      v 📁 mongo
          TS candidate-store.ts
          TS candidate.ts
          TS donation-store.ts
          TS donation.ts
          TS user-store.ts
          TS user.ts
        TS db.ts
    v 📁 services
        TS donation-service.ts
        TS donation-utils.ts
    v 📁 types
        TS donation-types.ts
```

Database Connection
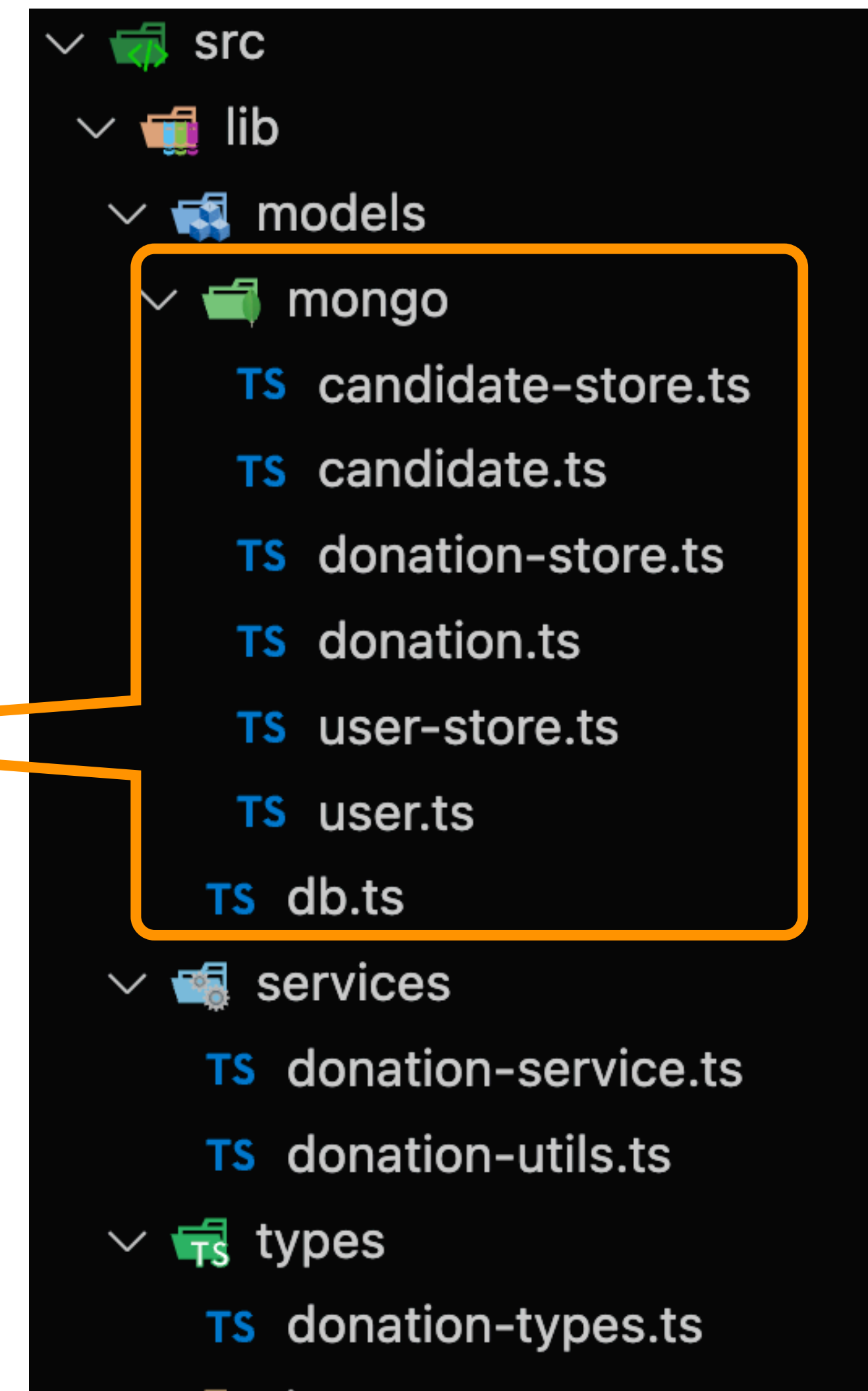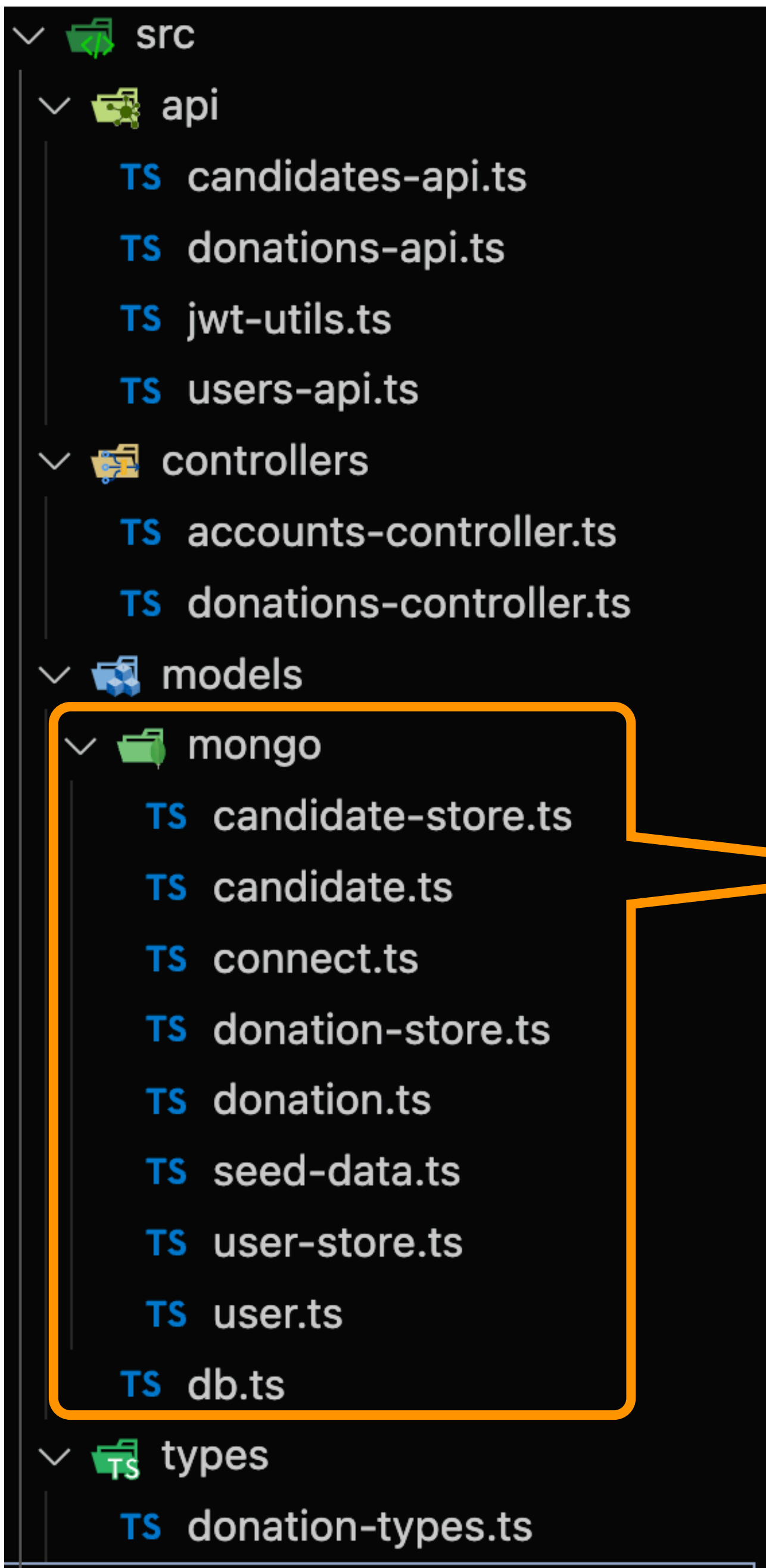
```javascript
import mongoose from "mongoose";
import { MONGO_URL } from "$env/static/private";
/*
  0 - disconnected
  1 - connected
  2 - connecting
  3 - disconnecting
  4 - uninitialized
*/
const mongoConnection = {
  isConnected: 0
};

export const dbConnect = async () => {
  console.log("MONGO_URL", MONGO_URL);
  if (mongoConnection.isConnected === 1) {
    console.log("already connected");
    return;
  }

  if (mongoose.connections.length > 0) {
    mongoConnection.isConnected = mongoose.connections[0].readyState;
    if (mongoConnection.isConnected === 1) {
      console.log("using existing connection");
      return;
    }

    await mongoose.disconnect();
  }
  await mongoose.connect(MONGO_URL);
  mongoConnection.isConnected = 1;
  console.log("conected to ", MONGO_URL ?? "");
};
```

- dbConnect() established connection of Mongo database server

- Only connect if not already connected

6
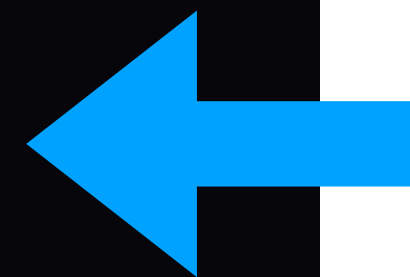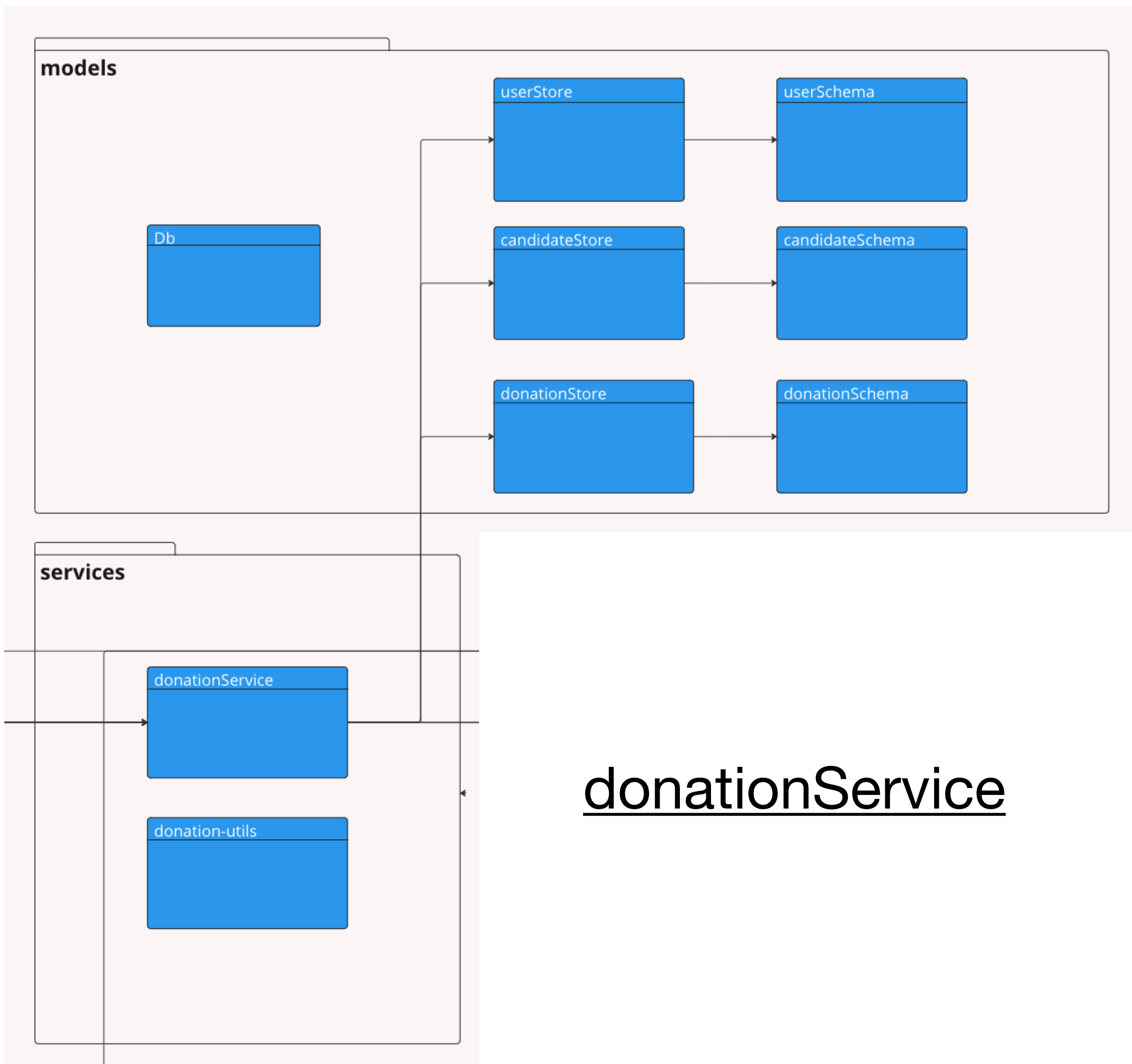
donationService

- Refactor donationService:

  - Remove API access via HTTP

  - Replace with direct access to the donation stores

```ts
export const donationService = {
  baseUrl: "http://localhost:4000",

  async login(email: string, password: string): Promise<Session | null> {
    try {
      const response = await axios.post(`${this.baseUrl}/api/users/authenticate`, { email, password });
      if (response.data.success) {
        axios.defaults.headers.common["Authorization"] = "Bearer " + response.data.token;
        const session: Session = {
          name: response.data.name,
          token: response.data.token,
          _id: response.data.id
        };
        return session;
      }
      return null;
    } catch (error) {
      console.log(error);
      return null;
    }
  },
```

- Access API over HTTP (axios)

```ts
export const donationService = {
  async login(email: string, password: string): Promise<Session | null> {
    try {
      const user = await userStore.findBy(email);
      if (user !== null && user.password === password) {
        const session = {
          name: `${user.firstName} ${user.lastName}`,
          token: user._id!.toString(),
          _id: user._id!.toString()
        };
        return session;
      }
      return null;
    } catch (error) {
      console.log(error);
      return null;
    }
  },
```

- Directly access userStore

donationService: login
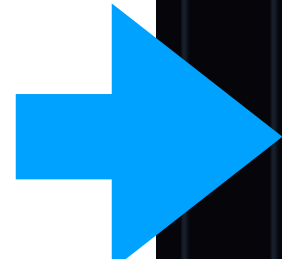
Donation-svelte-07-api

- Access API over HTTP (axios)

```
async donate(donation: Donation, session: Session) {
  try {
    axios.defaults.headers.common["Authorization"] = "Bearer " + session.token;
    const response = await axios.post(this.baseUrl + "/api/candidates/" + donation.candidate + "/donations", donation);
    return response.status == 200;
  } catch (error) {
    return false;
  }
},
```

Donation-svelte-08-full

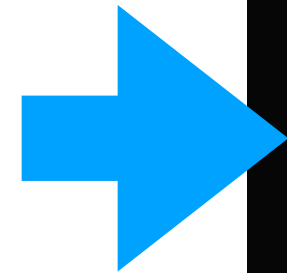- Directly access donationStore

```
async donate(donation: Donation) {
  try {
    donationStore.add(donation);
  } catch (error) {
    return false;
  }
},
```

donationService: donate

Hooks

- 'Hooks' are app-wide functions you declare that SvelteKit will call in response to specific events, giving you fine-grained control over the framework's behaviour.

- Connect to database on the server

```ts
import { dbConnect } from "$lib/models/db";
import type { Handle } from "@sveltejs/kit";


await dbConnect();

export const handle: Handle = async ({ event, resolve }) => {
  const response = await resolve(event);
  return response;
};
```

hooks.server.ts

Full Stack Implementation

Implementing SvelteKit full stack apps