

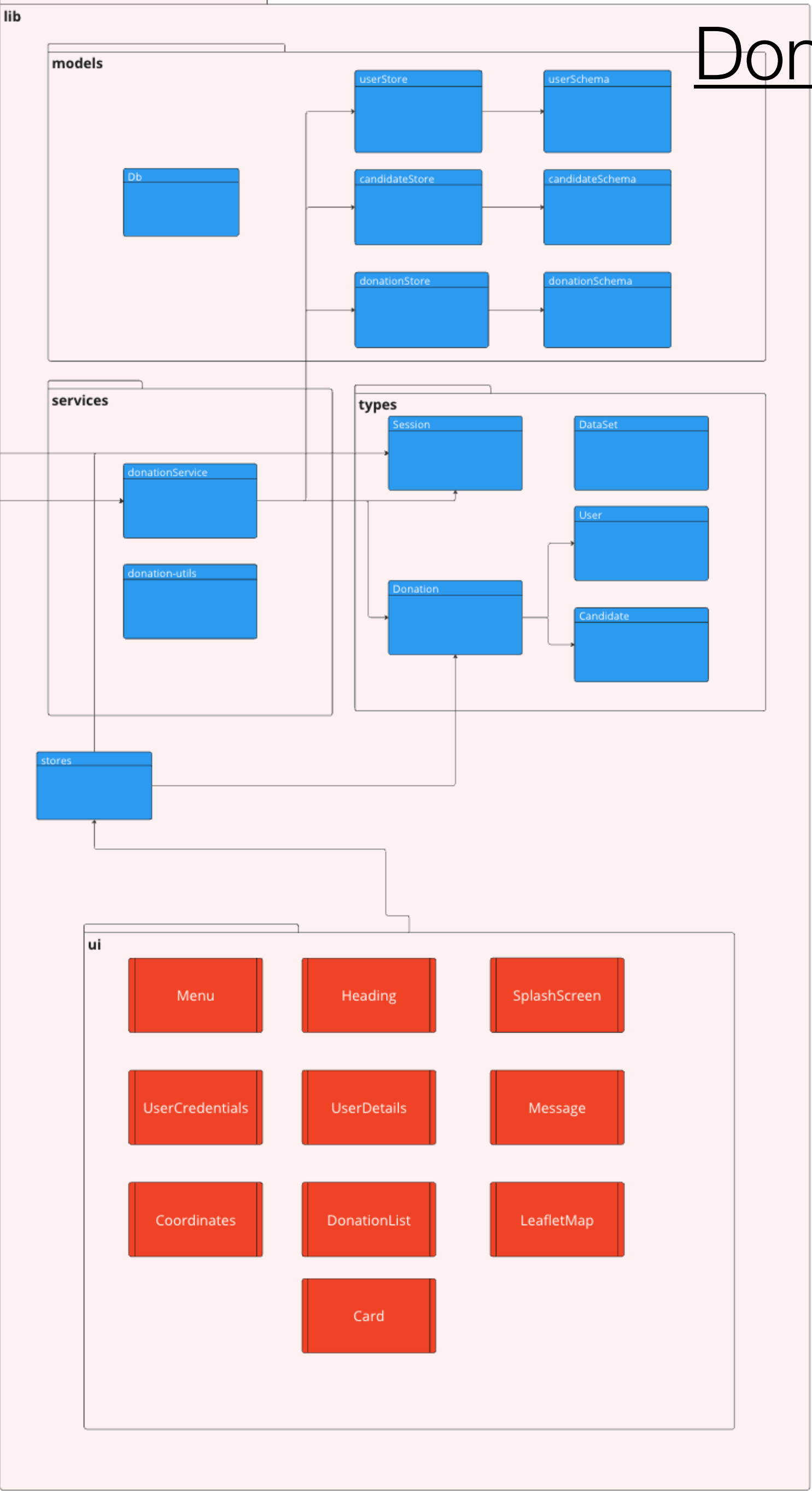
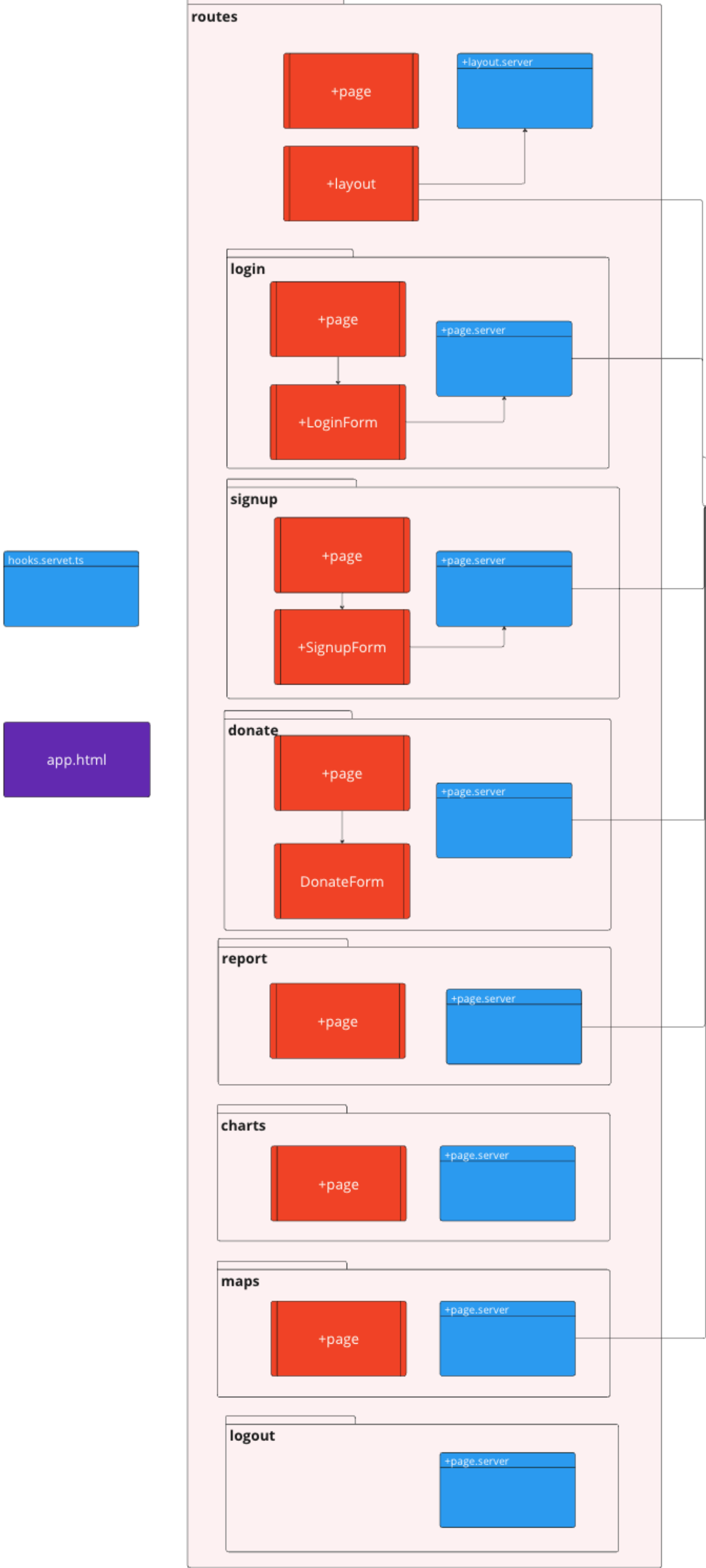
Full Stack Implementation



Implementing SvelteKit full
stack apps

Donation-svelte-09-full

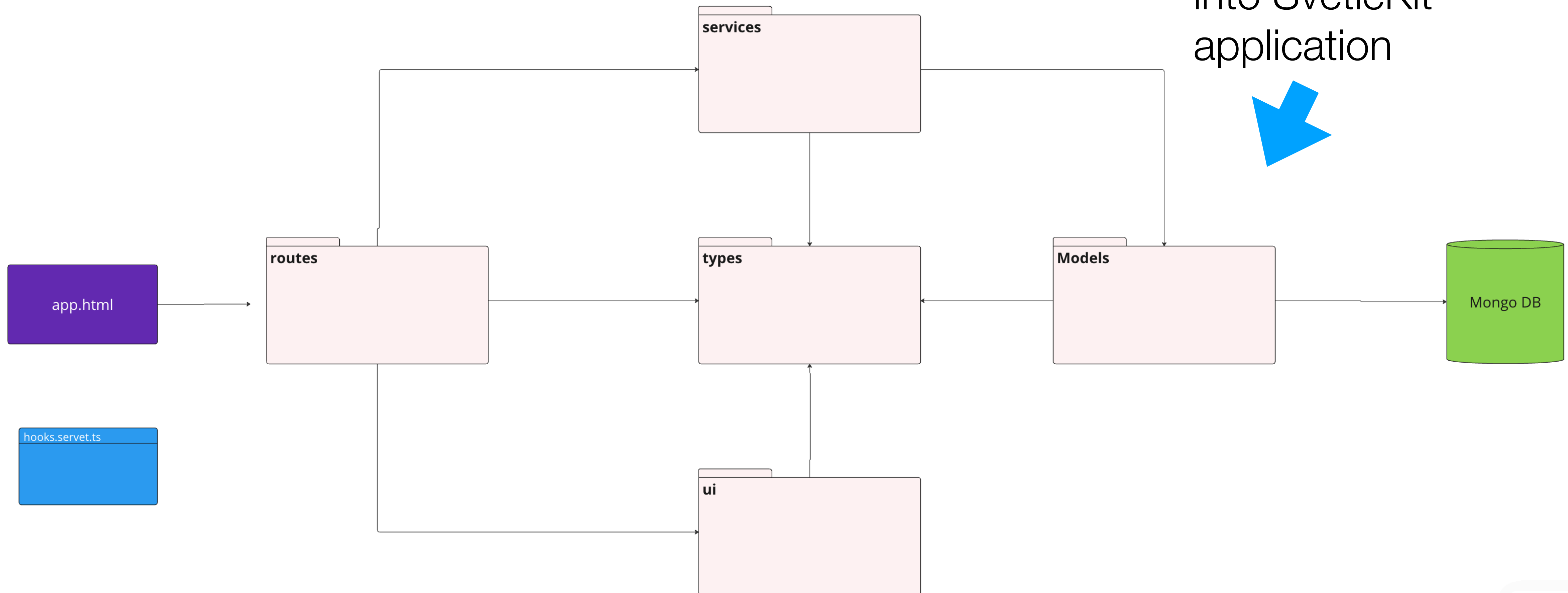
- routes
 - charts
 - TS +page.server.ts
 - +page.svelte
 - donate
 - TS +page.server.ts
 - +page.svelte
 - DonateForm.svelte
 - login
 - TS +page.server.ts
 - +page.svelte
 - LoginForm.svelte
 - logout
 - TS +page.server.ts
 - +page.svelte
 - maps
 - TS +page.server.ts
 - +page.svelte
 - report
 - TS +page.server.ts
 - +page.svelte
 - signup
 - +page.svelte
 - SignupForm.svelte
 - TS +layout.server.ts
 - +layout.svelte
 - +page.svelte
 - app.d.ts
 - app.html
 - TS hooks.server.ts



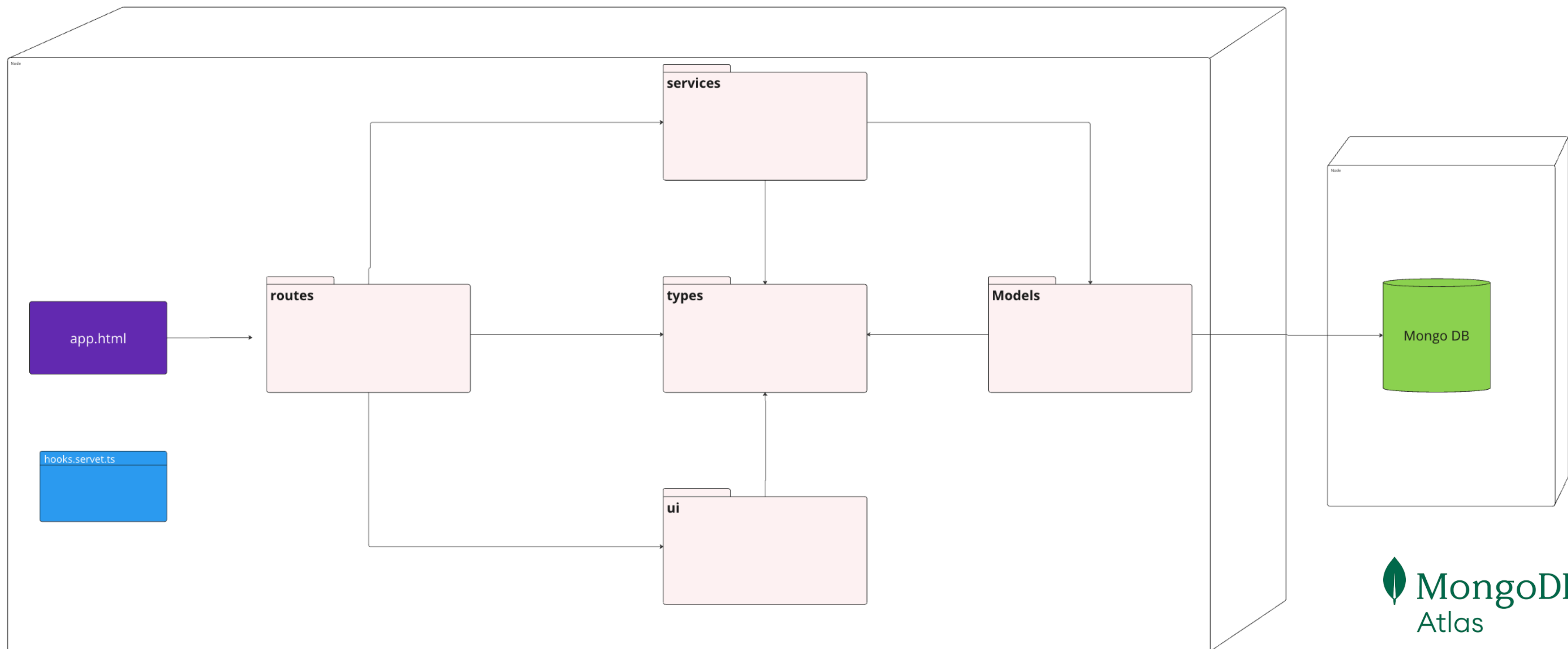
- src
 - lib
 - models
 - TS candidate-store.ts
 - TS candidate.ts
 - TS donation-store.ts
 - TS donation.ts
 - TS user-store.ts
 - TS user.ts
 - TS db.ts
 - services
 - TS donation-service.ts
 - TS donation-utils.ts
 - types
 - TS donation-types.ts
 - ui
 - Card.svelte
 - Coordinates.svelte
 - DonationList.svelte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

Donation Full Stack Architecture

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



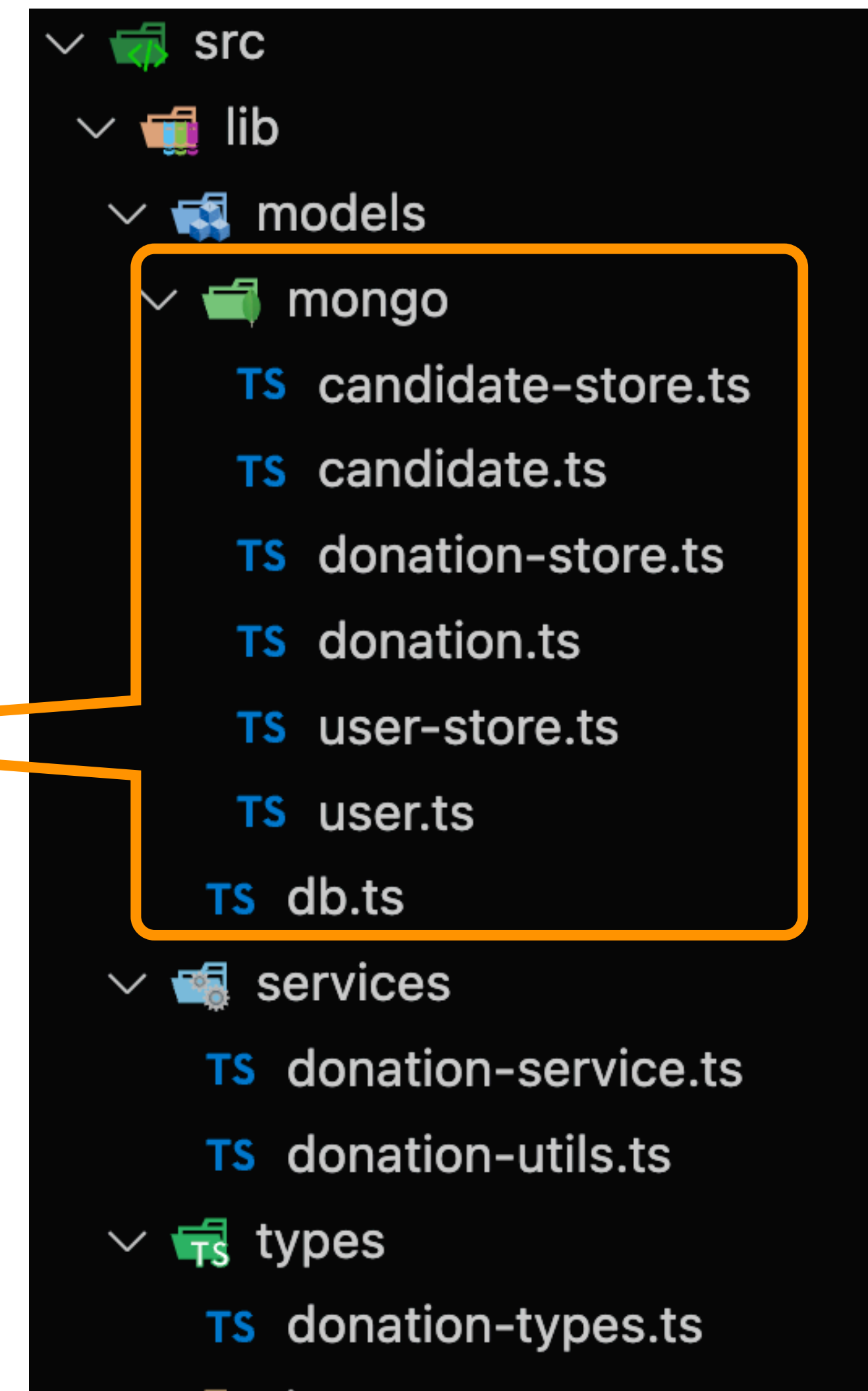
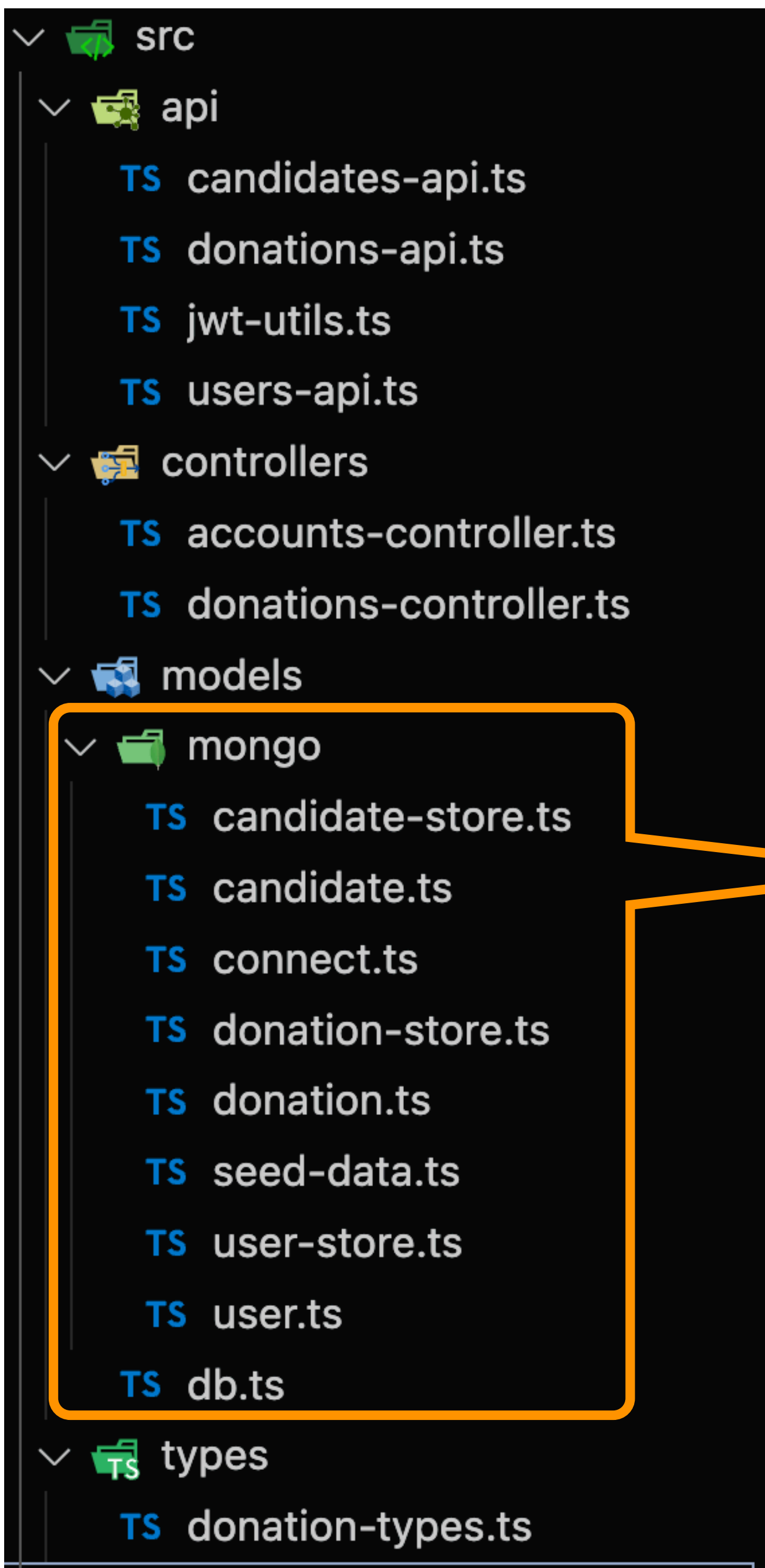
Donation Full Stack Deployment



Migrate Mongoose Models

- From Donation-hapi app
- ... to SvelteKit app

```
npm install mongoose
```



Database Connection

```
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
};

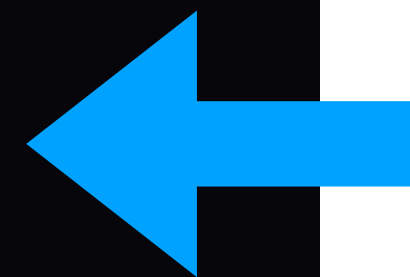
export const dbConnect = async () => {
  console.log("MONGO_URL", MONGO_URL);
  if (mongoConnection.isConnected === 1) {
    console.log("already connected");
    return;
  }

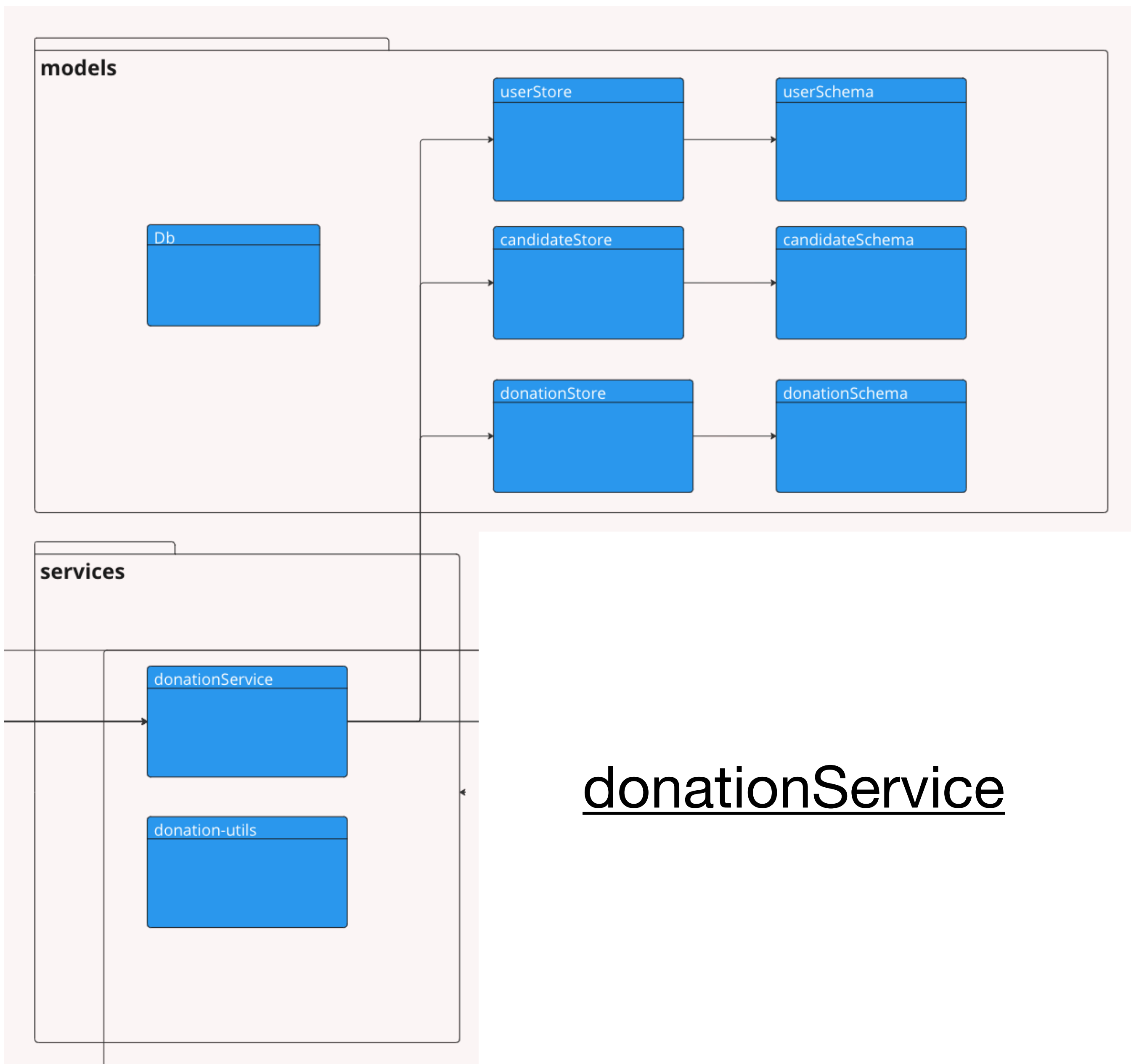
  if (mongoose.connections.length > 0) {
    mongoConnection.isConnected = mongoose.connections[0].readyState;
    if (mongoConnection.isConnected === 1) {
      console.log("using existing connection");
      return;
    }

    await mongoose.disconnect();
  }

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

- dbConnect() established connection of Mongo database server
- Only connect if not already connected

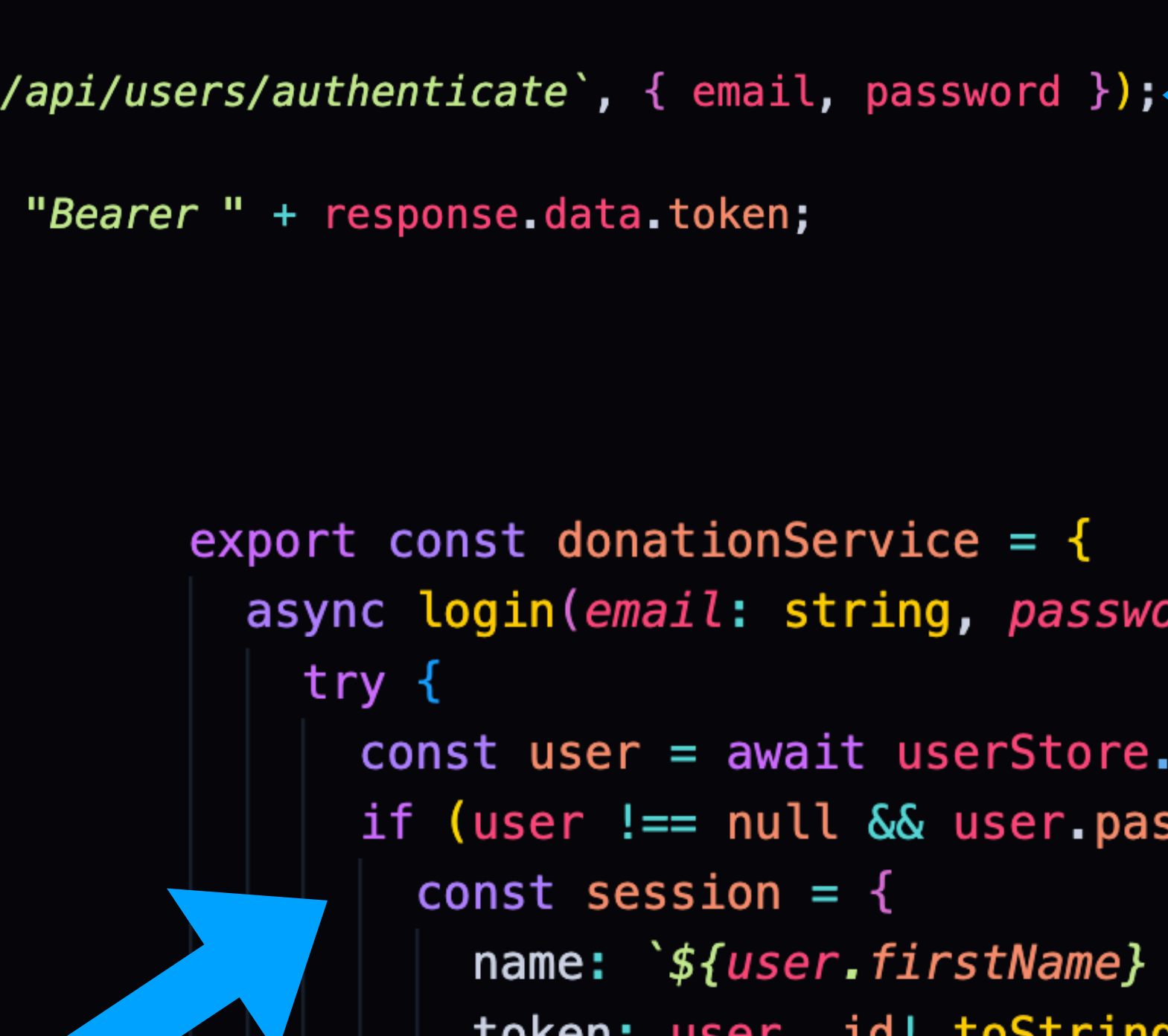




- Refactor donationService:
 - Remove API access via HTTP
 - Replace with direct access to the donation stores


```
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;
    }
  },
};
```



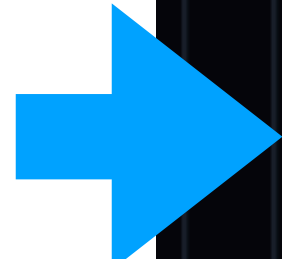
```
export const donationService = {
  async login(email: string, password: string): Promise<Session | null> {
    try {
      const user = await userStore.findBy(email);
      if (user !== null && user.password === password) {
        const session = {
          name: `${user.firstName} ${user.lastName}`,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

- Directly access userStore



```
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;  
  }  
},
```

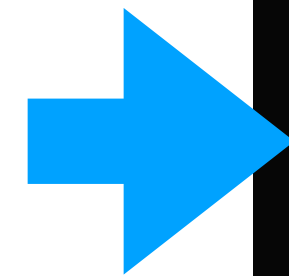
- Directly access donationStore



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

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.



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

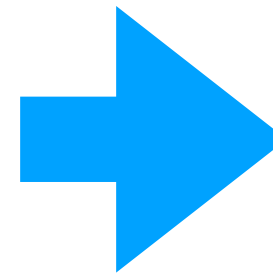
await dbConnect();

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

hooks.server.ts

Mongoose Model Adjustments

Model in Hapi

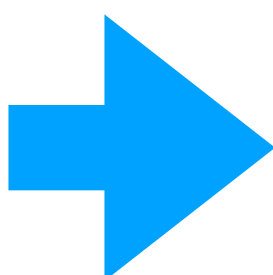


```
import { Schema, model } from "mongoose";
import { Candidate } from "../../types/donation-types";

const candidateSchema = new Schema<Candidate>({
  firstName: String,
  lastName: String,
  office: String,
});

export const CandidateMongoose = model("Candidate", candidateSchema);
```

Adjusted model for
SvelteKit



```
import type { Candidate } from "$lib/types/donation-types";
import mongoose, { Model } from "mongoose";

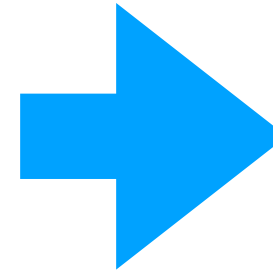
const candidateSchema = new mongoose.Schema<Candidate>({
  firstName: String,
  lastName: String,
  office: String
});

let CandidateMongoose: Model<Candidate>;
try {
  CandidateMongoose = mongoose.model<Candidate>("Candidate");
} catch {
  CandidateMongoose = mongoose.model<Candidate>("Candidate", candidateSchema);
}

export { CandidateMongoose };
```

Mongoose Model Adjustments

Model in Hapi

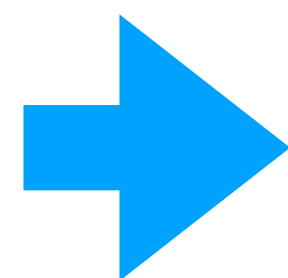


```
import { Schema, model } from "mongoose";
import { Candidate } from "../../types/donation-types";

const candidateSchema = new Schema<Candidate>({
  firstName: String,
  lastName: String,
  office: String,
});

export const CandidateMongoose = model("Candidate", candidateSchema);
```

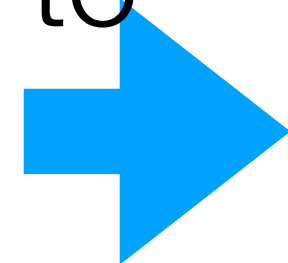
Adjusted model for SvelteKit



```
import type { Candidate } from "$lib/types/donation-types";
import mongoose, { Model } from "mongoose";

const candidateSchema = new mongoose.Schema<Candidate>({
  firstName: String,
  lastName: String,
  office: String
});
```

- Due to HMR (Hot Module Reload) behaviour, need to reload the mongoose schema if it has been unloaded



```
let CandidateMongoose: Model<Candidate>;
try {
  CandidateMongoose = mongoose.model<Candidate>("Candidate");
} catch {
  CandidateMongoose = mongoose.model<Candidate>("Candidate", candidateSchema);
}

export { CandidateMongoose };
```


Full Stack Implementation



Implementing SvelteKit full
stack apps