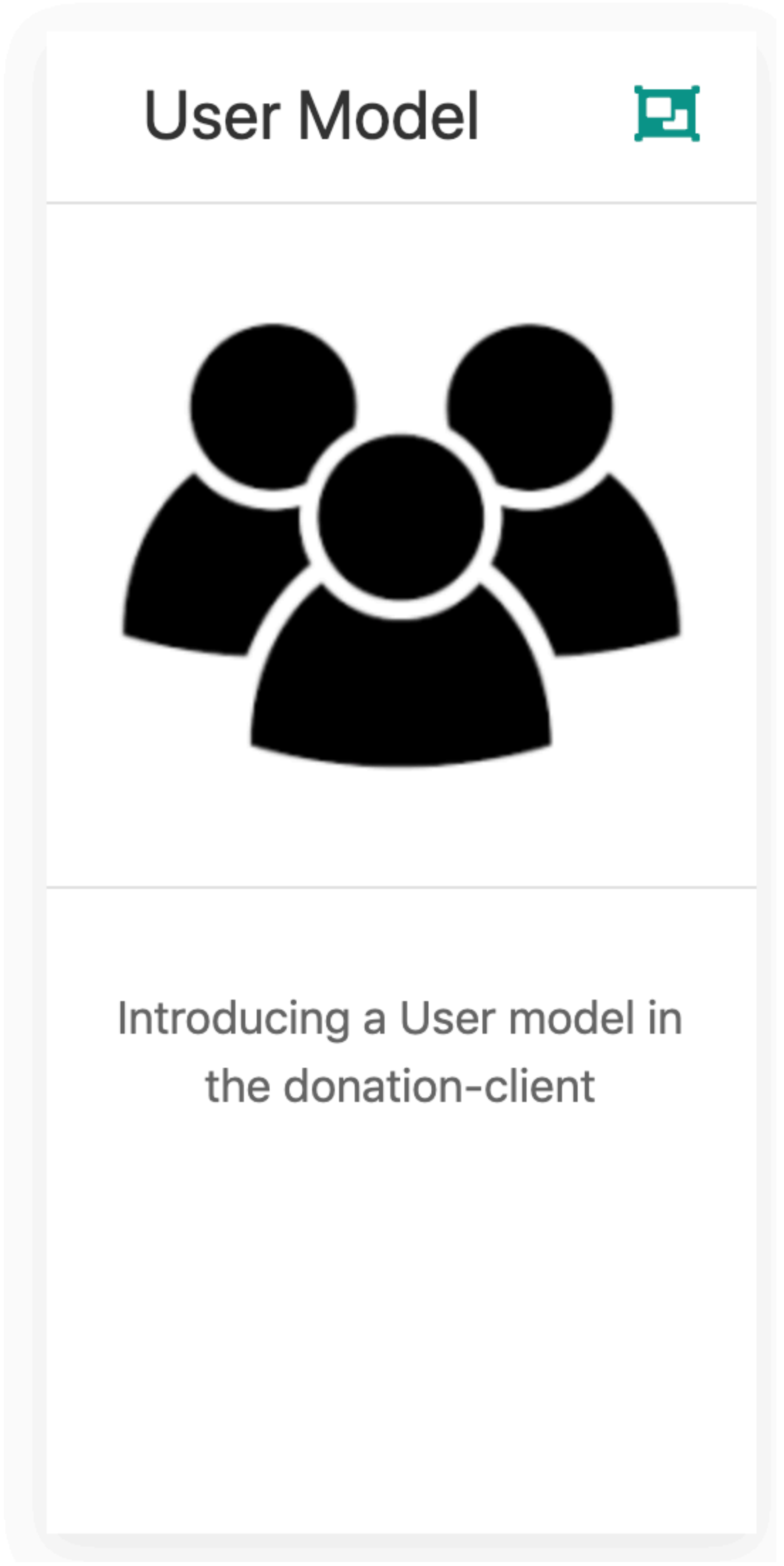
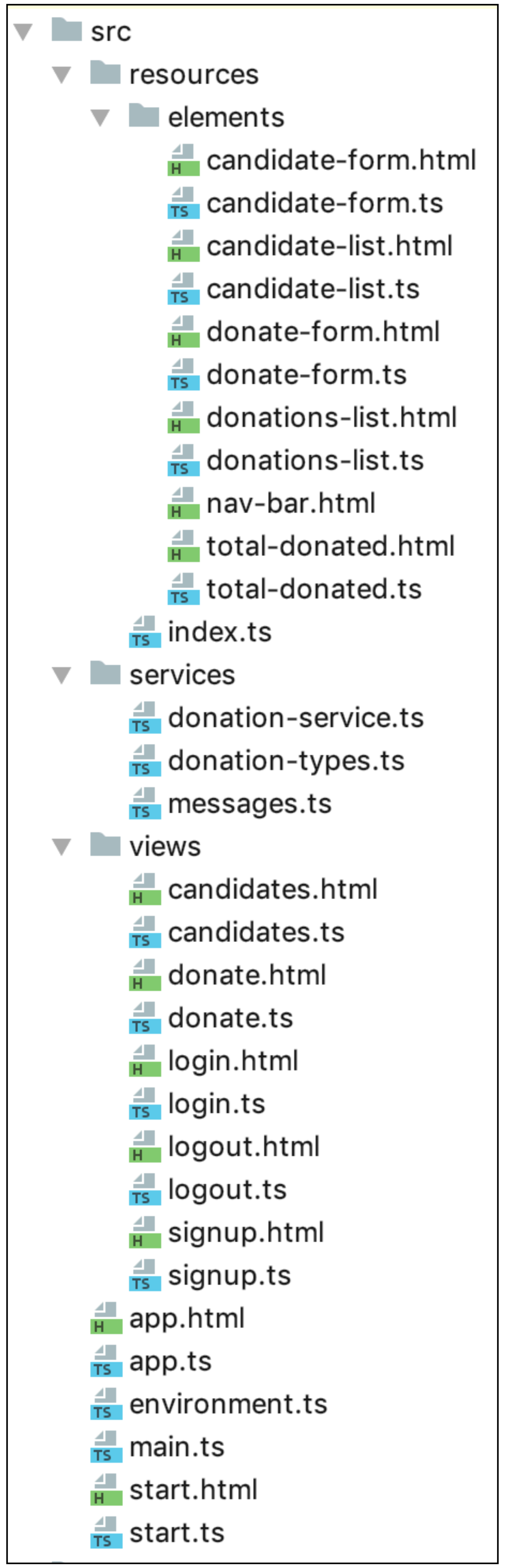
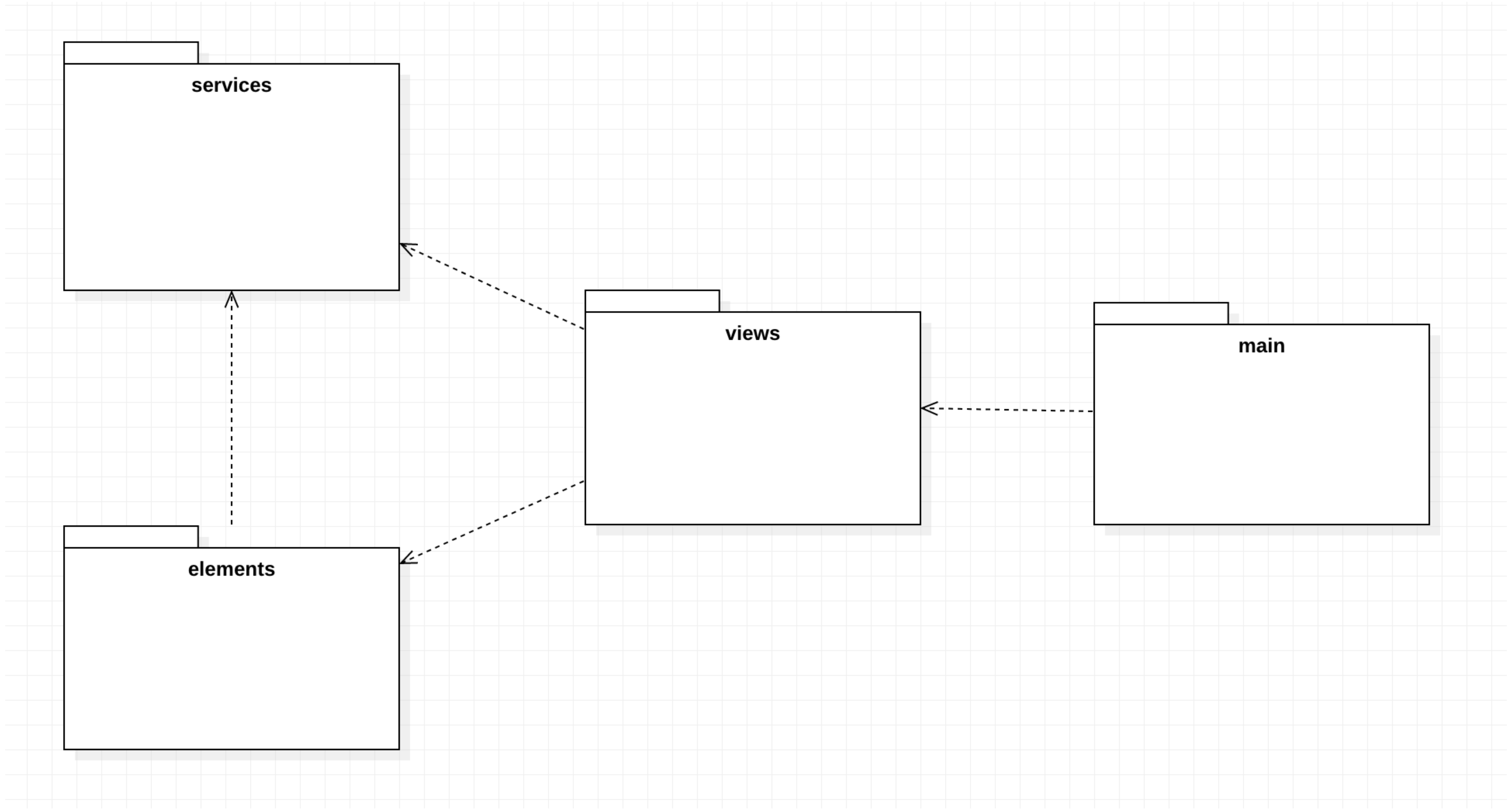
User Model


User Model

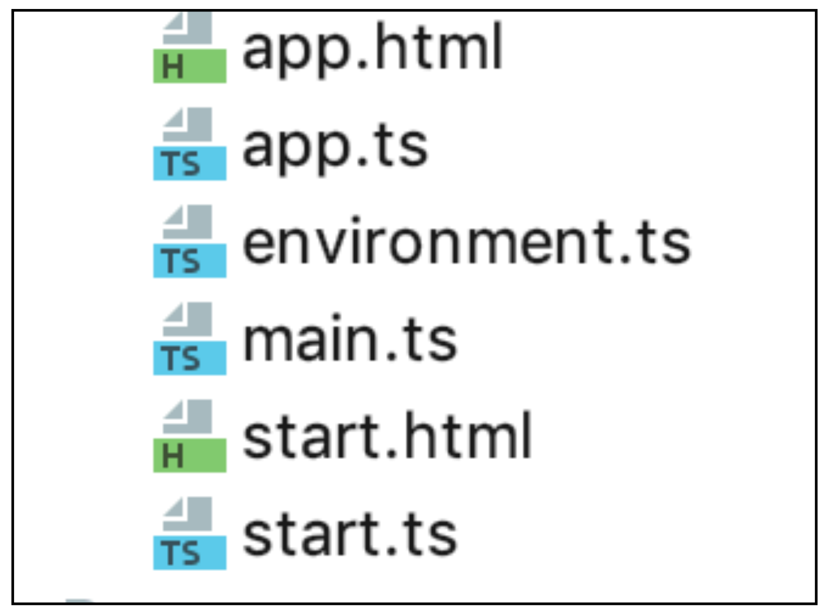
Introducing a User model in the donation-client

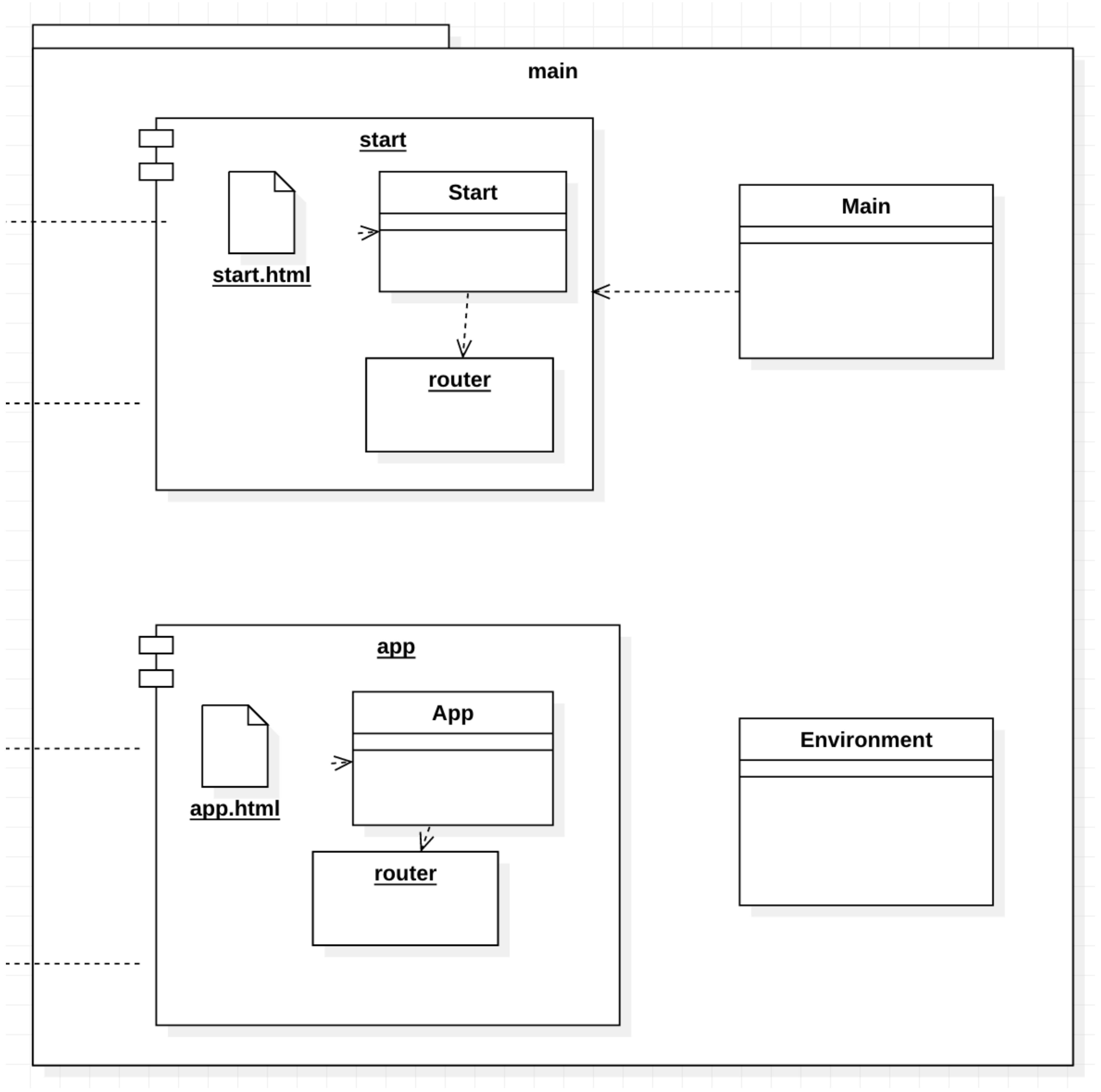


```
src
  resources
    elements
      candidate-form.html
      candidate-form.ts
      candidate-list.html
      candidate-list.ts
      donate-form.html
      donate-form.ts
      donations-list.html
      donations-list.ts
      nav-bar.html
      total-donated.html
      total-donated.ts
    index.ts
  services
    donation-service.ts
    donation-types.ts
    messages.ts
  views
    candidates.html
    candidates.ts
    donate.html
    donate.ts
    login.html
    login.ts
    logout.html
    logout.ts
    signup.html
    signup.ts
  app.html
  app.ts
  environment.ts
  main.ts
  start.html
  start.ts
```



main


app.html
app.ts
environment.ts
main.ts
start.html
start.ts

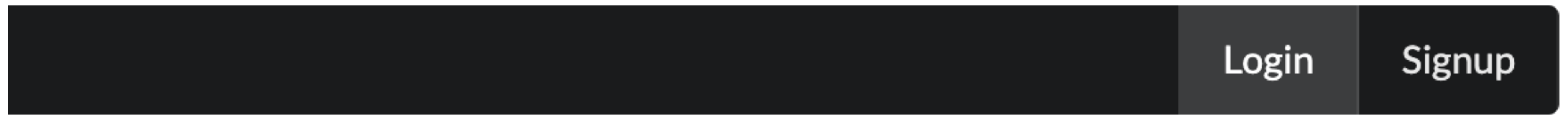


Define 2 routers

- **start**: support signup/ login

- **app**: support donation/ candidates (authenticated user)

start.ts

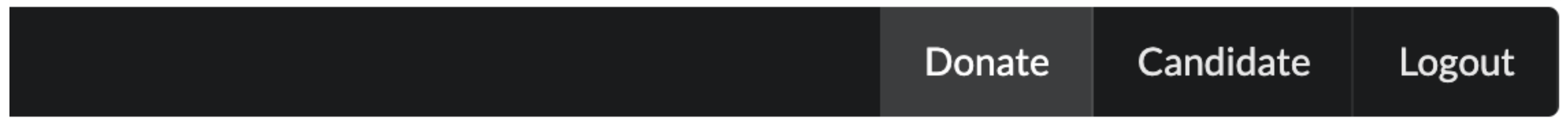


```
export class Start {
  router: Router;

  configureRouter(config: RouterConfiguration, router: Router) {
    config.map([
      {
        route: ['', 'login'],
        name: 'Login',
        moduleId: PLATFORM.moduleName('views/login'),
        nav: true,
        title: 'Login'
      },
      {
        route: 'signup',
        name: 'signup',
        moduleId: PLATFORM.moduleName('views/signup'),
        nav: true,
        title: 'Signup'
      }
    ]);
    this.router = router;
  }
}
```

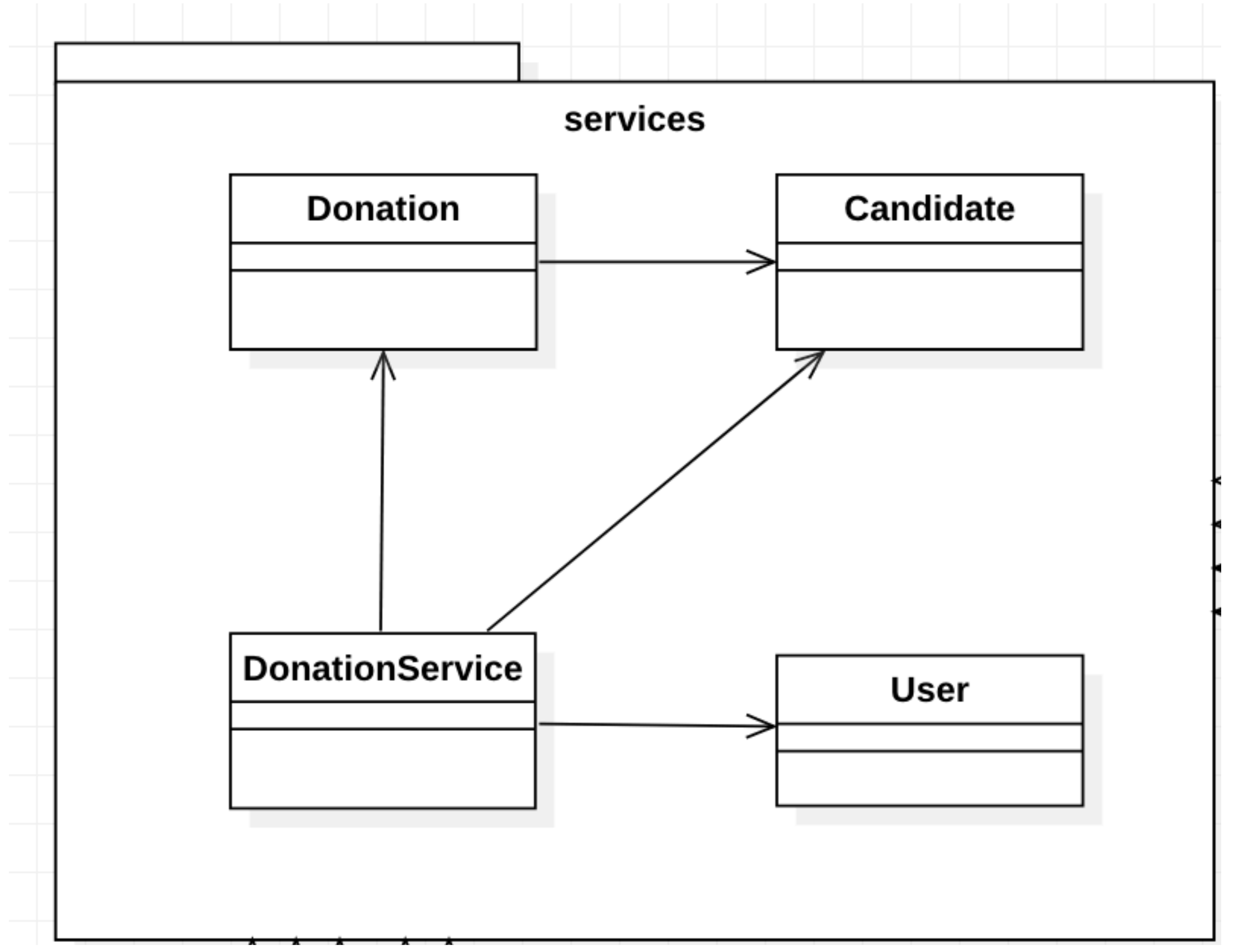
start/app.html

```
<template>
  <require from="resources/elements/nav-bar.html"></require>
  <div class="ui container">
    <section class="ui raised segment">
      <h3 class="ui headder"> Donation </h3>
    </section>
    <nav-bar router.bind="router"></nav-bar>
    <div class="ui basic segment">
      <router-view></router-view>
    </div>
  </div>
</template>
```

app.ts



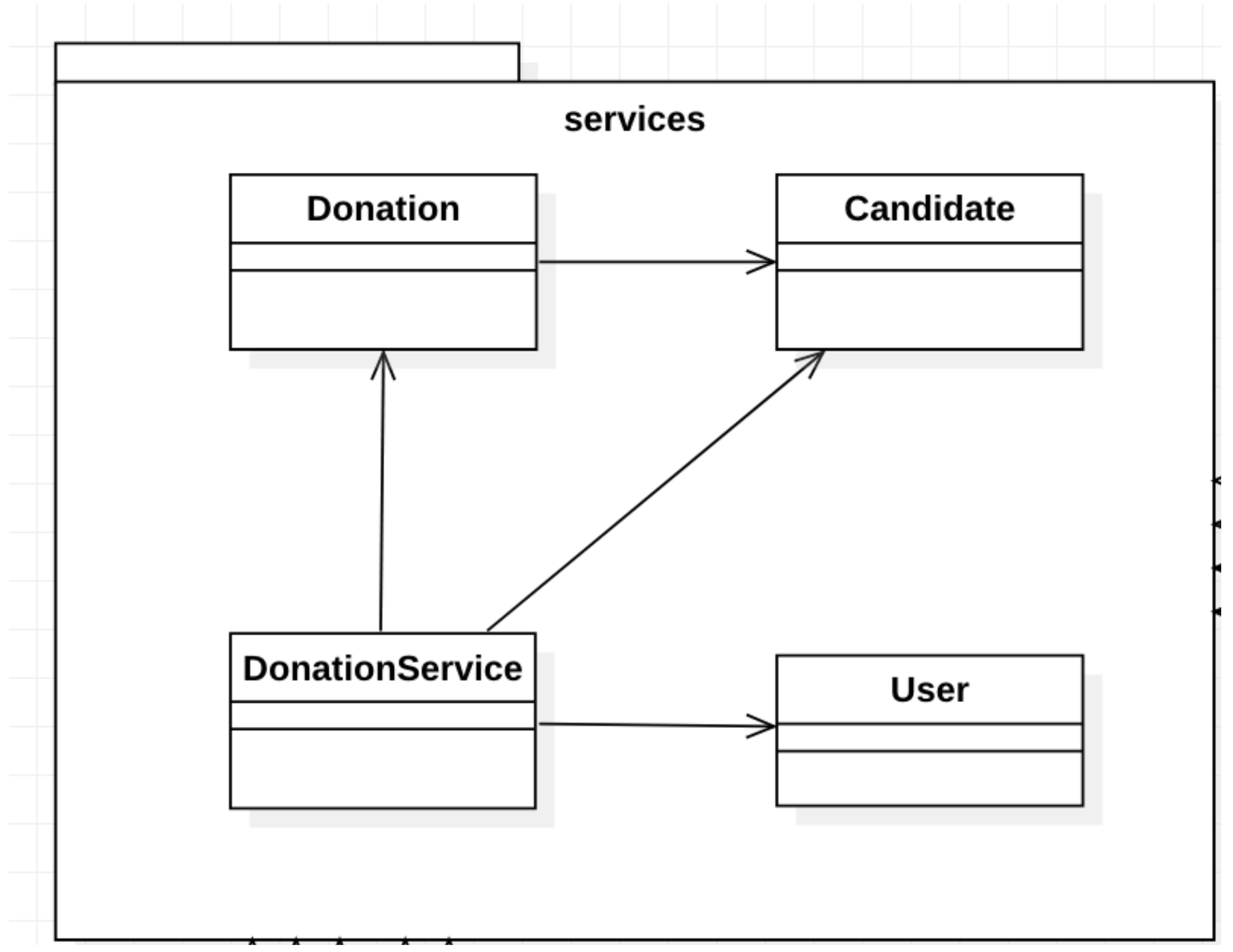
```
export class App {
  router: Router;

  configureRouter(config: RouterConfiguration, router: Router) {
    config.map([
      {
        route: ['', 'donate'],
        name: 'Donate',
        moduleId: PLATFORM.moduleName('views/donate'),
        nav: true,
        title: 'Donate'
      },
      {
        route: 'candidates',
        name: 'candidates',
        moduleId: PLATFORM.moduleName('views/candidates'),
        nav: true,
        title: 'Candidate'
      },
      {
        route: 'logout',
        name: 'logout',
        moduleId: PLATFORM.moduleName('views/logout'),
        nav: true,
        title: 'Logout'
      }
    ]);
    this.router = router;
  }
}
```



donation-types.ts

```
export interface Candidate {
  firstName: string;
  lastName: string;
  office: string;
}

export interface Donation {
  amount: number;
  method: string;
  candidate: Candidate;
}

export interface User {
  firstName: string;
  lastName: string;
  email: string;
  password: string;
}
```

donation-service.ts

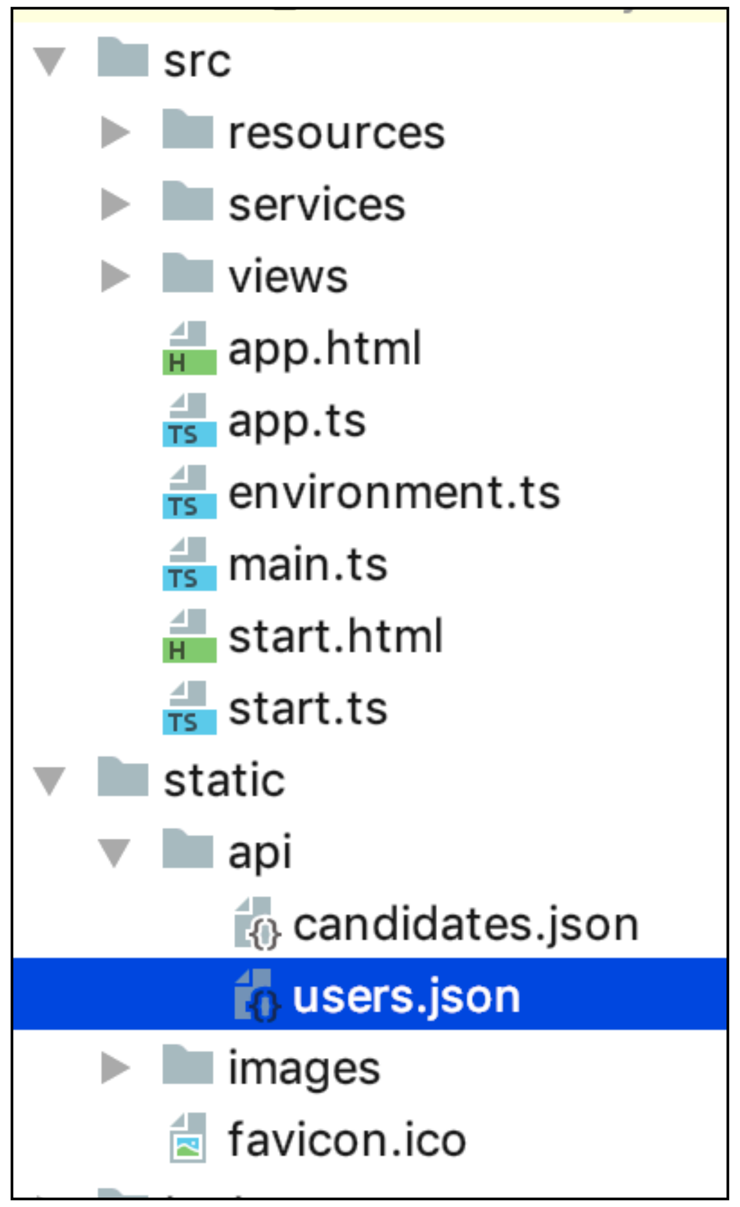


```
@inject(HttpClient, EventAggregator, Aurelia, Router)
export class DonationService {
  users: Map<string, User> = new Map();
  candidates: Candidate[] = [];
  donations: Donation[] = [];
  paymentMethods = ['Cash', 'Paypal'];
  total = 0;

  constructor(private httpClient: HttpClient, private ea: EventAggregator,
              private au: Aurelia,            private router: Router) {
    httpClient.configure(http => {
      http.withBaseUrl('http://localhost:8080');
    });
    this.getCandidates();
    this.getUsers();
  }

  async getCandidates() {
    const response = await this.httpClient.get('/api/candidates.json');
    this.candidates = await response.content;
  }

  async getUsers() {
    const response = await this.httpClient.get('/api/users.json');
    const users = await response.content;
    users.forEach(user => {
      this.users.set(user.email, user);
    });
  }
```

donation-service.ts: users

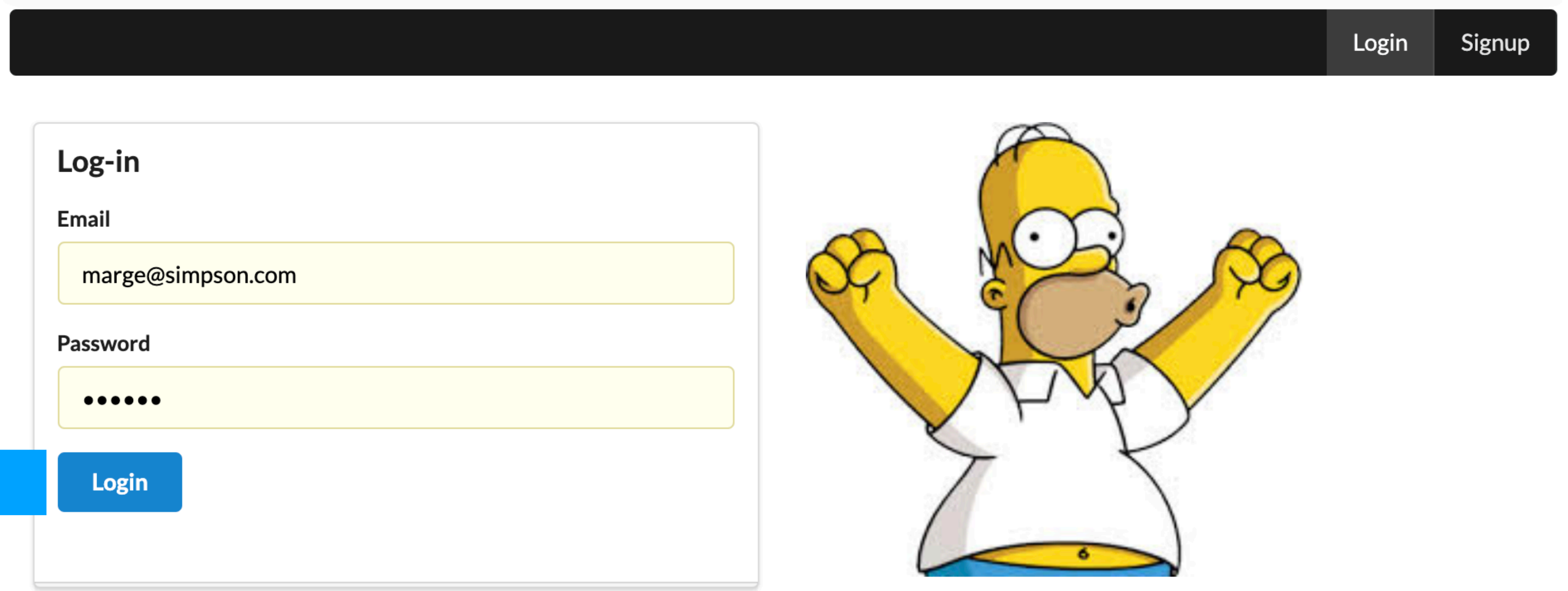
```
export class DonationService {

  users: Map<string, User> = new Map();

  async getUsers() {
    const response = await this.httpClient.get('/api/users.json');
    const users = await response.content;
    users.forEach(user => {
      this.users.set(user.email, user);
    });
  }

  async login(email: string, password: string) {
    const user = this.users.get(email);
    if (user && (user.password === password)) {
      this.changeRouter(PLATFORM.moduleName('app'))
      return true;
    } else {
      return false;
    }

  ...

}
```

Test users loaded from 'users.json' file



```
[
  {
    "firstName": "Homer",
    "lastName": "Simpson",
    "email": "homer@simpson.com",
    "password": "secret"
  },
  {
    "firstName": "Marge",
    "lastName": "Simpson",
    "email": "marge@simpson.com",
    "password": "secret"
  },
  {
    "firstName": "Bart",
    "lastName": "Simpson",
    "email": "bart@simpson.com",
    "password": "secret"
  }
]
```

donation-service.ts: login


Login
Signup

Log-in

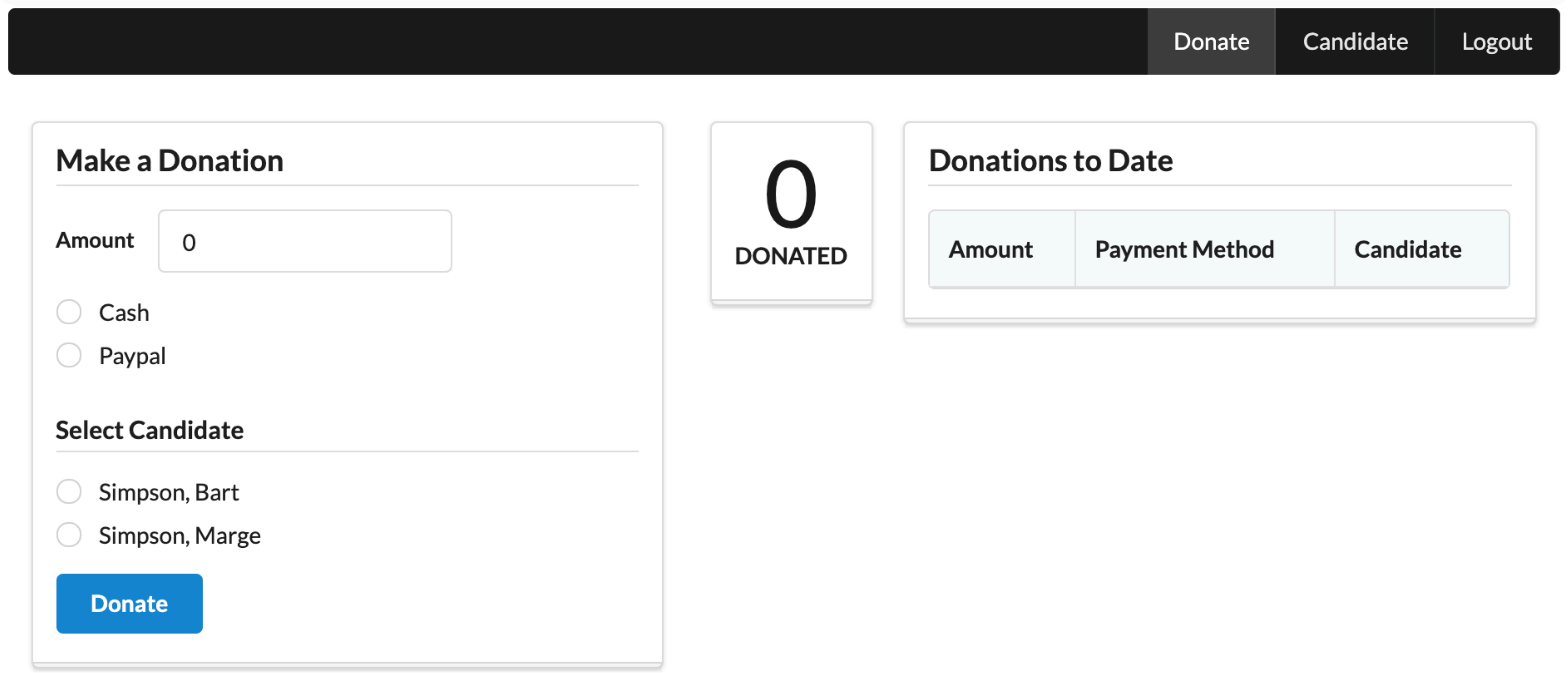
Email
marge@simpson.com

Password
••••••

Login

```
export class DonationService {


  async login(email: string, password: string) {
    const user = this.users.get(email);
    if (user && (user.password === password)) {
      this.changeRouter(PLATFORM.moduleName('app'))
      return true;
    } else {
      return false;
    }

  ...

}
```

If user found - change the router


Donate
Candidate
Logout

Make a Donation

Amount 0

Cash
Paypal

Select Candidate

Simpson, Bart
Simpson, Marge

Donate

0
DONATED

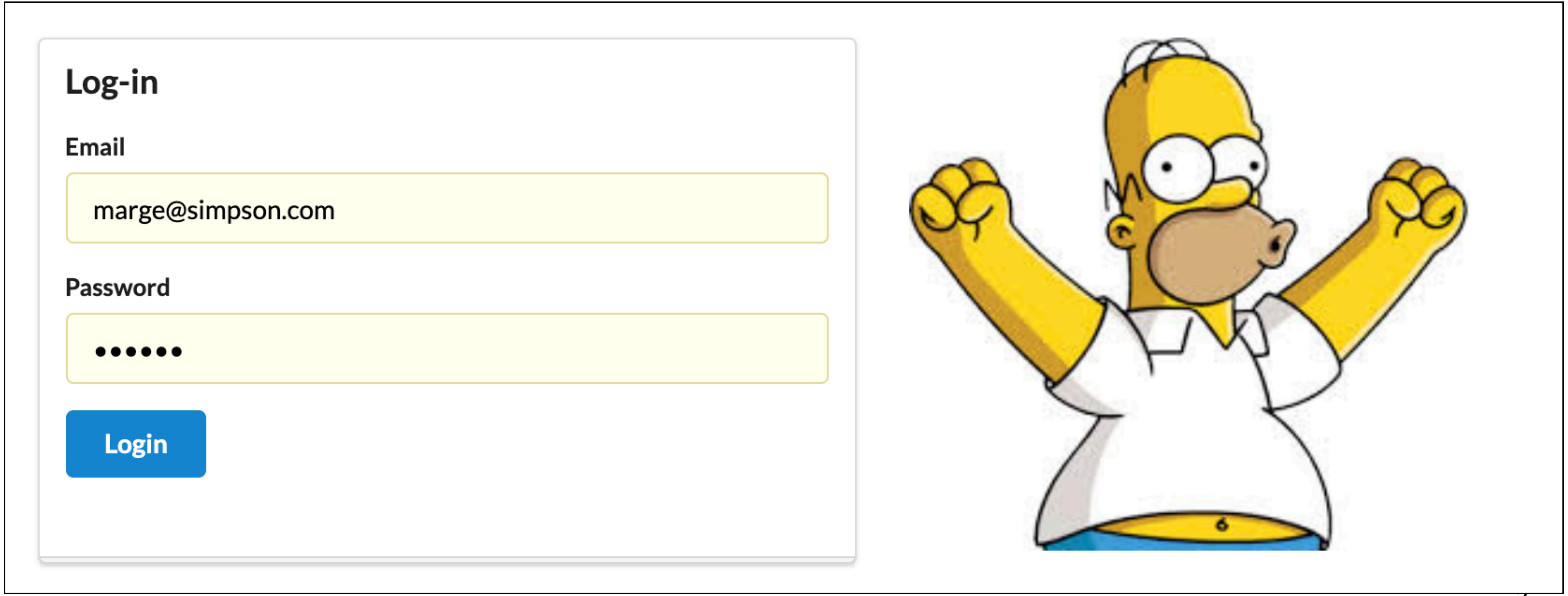
Donations to Date

Amount	Payment Method	Candidate

login View Component

```
@inject(DonationService)
export class Login {
  email = 'marge@simpson.com';
  password = 'secret';
  prompt = '';

  constructor(private ds: DonationService) {}

  async login(e) {
    console.log(`Trying to log in ${this.email}`);
    const success = await this.ds.login(this.email, this.password);
    if (!success) {
      this.prompt = "Oops! Try again...";
    }
  }
}
```

```
<template>
  <div class="ui stackable two column grid">
    <div class="column">
      <form submit.delegate="login($event)" class="ui stacked segment form">
        <h3 class="ui header">Log-in</h3>
        <div class="field">
          <label>Email</label> <input placeholder="Email" value.bind="email"/>
        </div>
        <div class="field">
          <label>Password</label> <input type="password" value.bind="password"/>
        </div>
        <button class="ui blue submit button">Login</button>
        <h3>${prompt}</h3>
      </form>
    </div>
    <div class="column">
      <img class="ui medium image" src="/images/homer4.jpeg">
    </div>
  </div>
</template>
```



Log-in

Email

marge@simpson.com

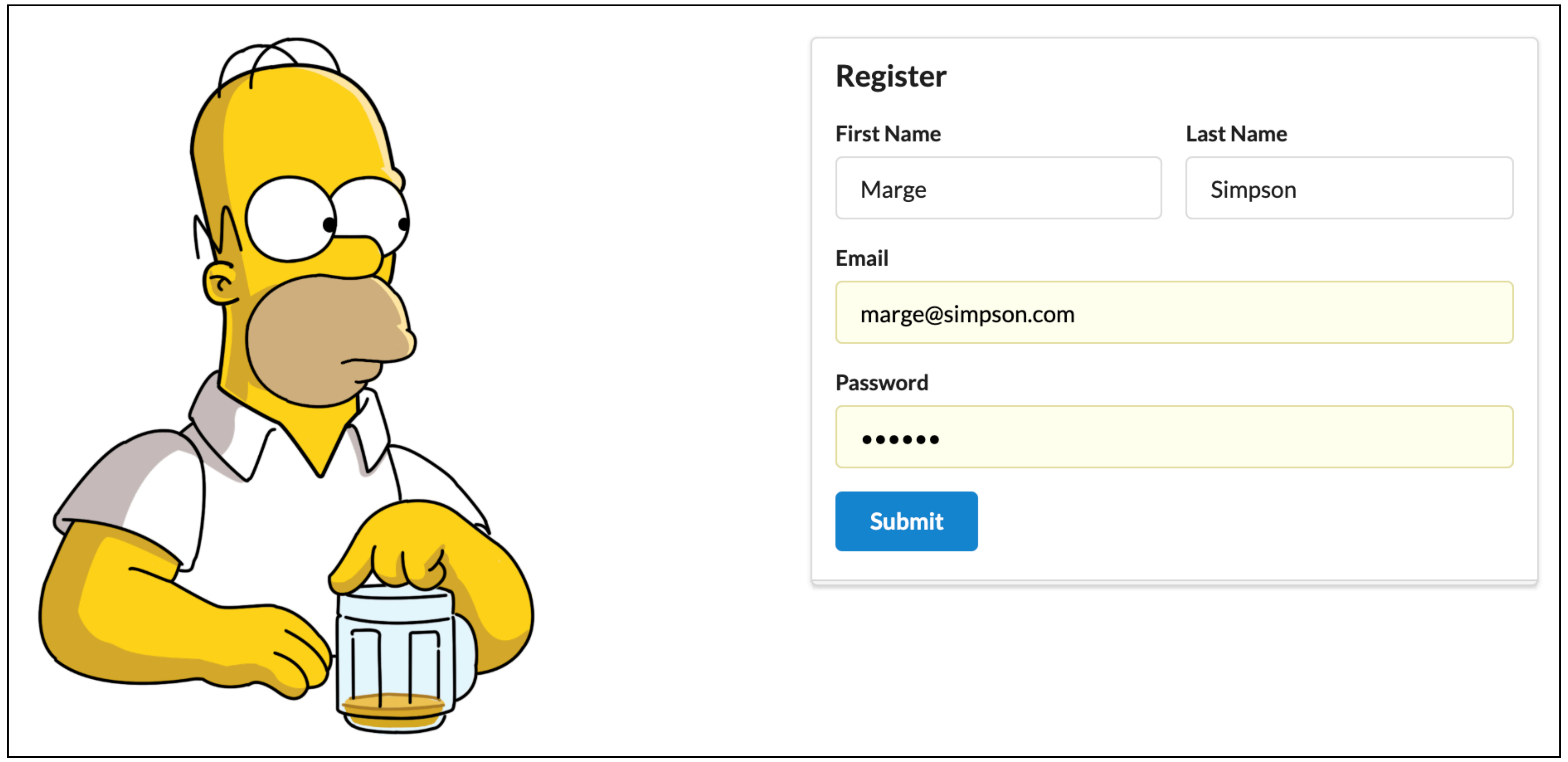
Password

••••••

signup View Component

```
@inject(DonationService)
export class Signup {
  firstName = 'Marge';
  lastName = 'Simpson';
  email = 'marge@simpson.com';
  password = 'secret';
  prompt = '';

  constructor(private ds: DonationService) {}

  signup(e) {
    console.log(`Trying to sign up ${this.email}`);
    const success = this.ds.signup(this.firstName, this.lastName,
                                   this.email, this.password);
    if (!success) {
      this.prompt = 'Oops! Try again...';
    }
  }
}
```

```
<template>
  <div class="ui stackable two column grid">
    <div class="column">
      <img class="ui medium image" src="/images/homer2.png">
    </div>
    <div class="column">
      <form submit.delegate="signup($event)" class="ui stacked segment form">
        <h3 class="ui header">Register</h3>
        <div class="two fields">
          <div class="field">
            <label>First Name</label> <input placeholder="First Name" type="text" value.bind="firstName">
          </div>
          <div class="field">
            <label>Last Name</label> <input placeholder="Last Name" type="text" value.bind="lastName">
          </div>
        </div>
        <div class="field">
          <label>Email</label> <input placeholder="Email" type="text" value.bind="email">
        </div>
        <div class="field">
          <label>Password</label> <input type="password" value.bind="password">
        </div>
        <button class="ui blue submit button">Submit</button>
      </form>
      <h3>${prompt}</h3>
    </div>
  </div>
</template>
```


Register

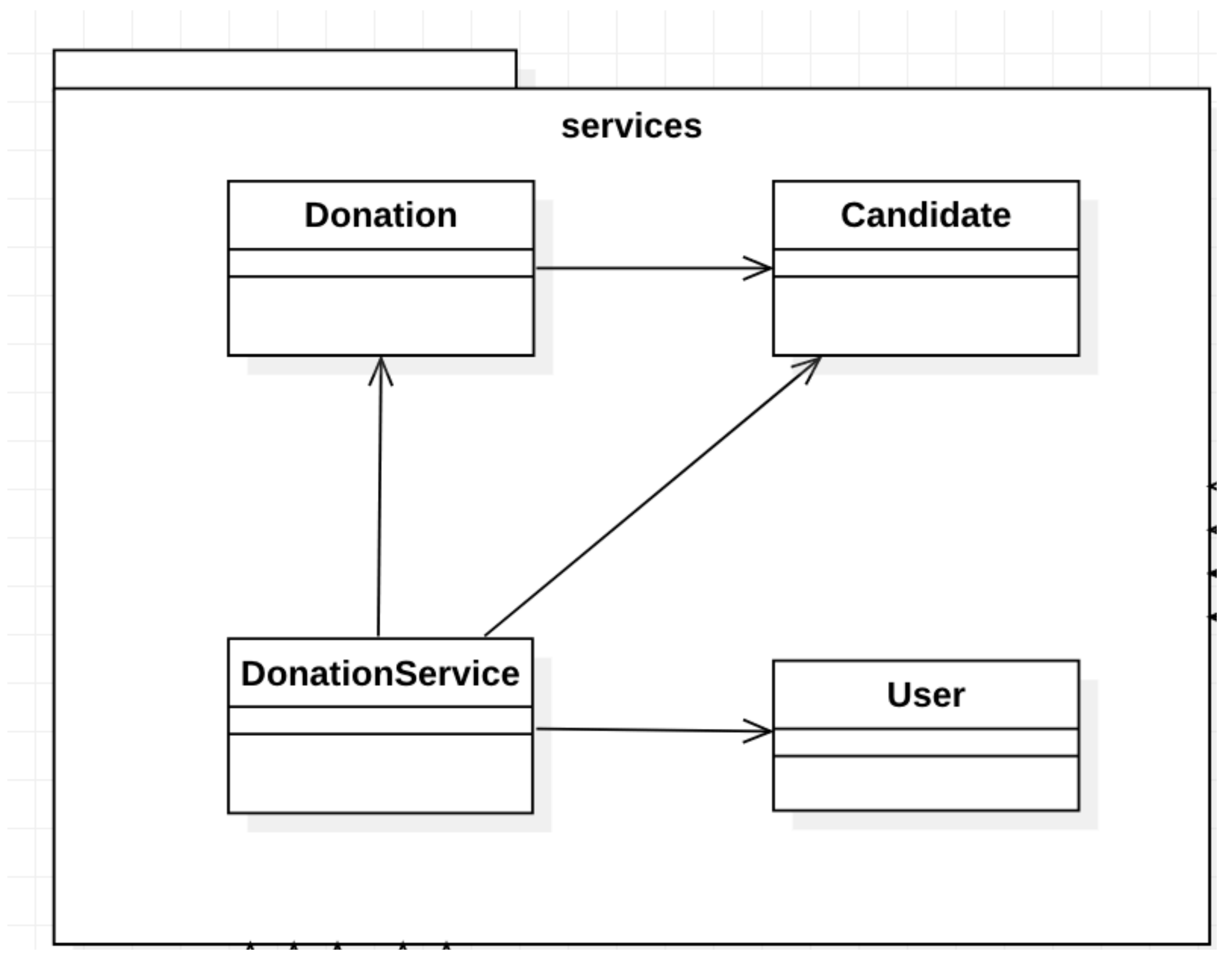
First Name: Marge

Last Name: Simpson

Email: marge@simpson.com

Password: ••••••

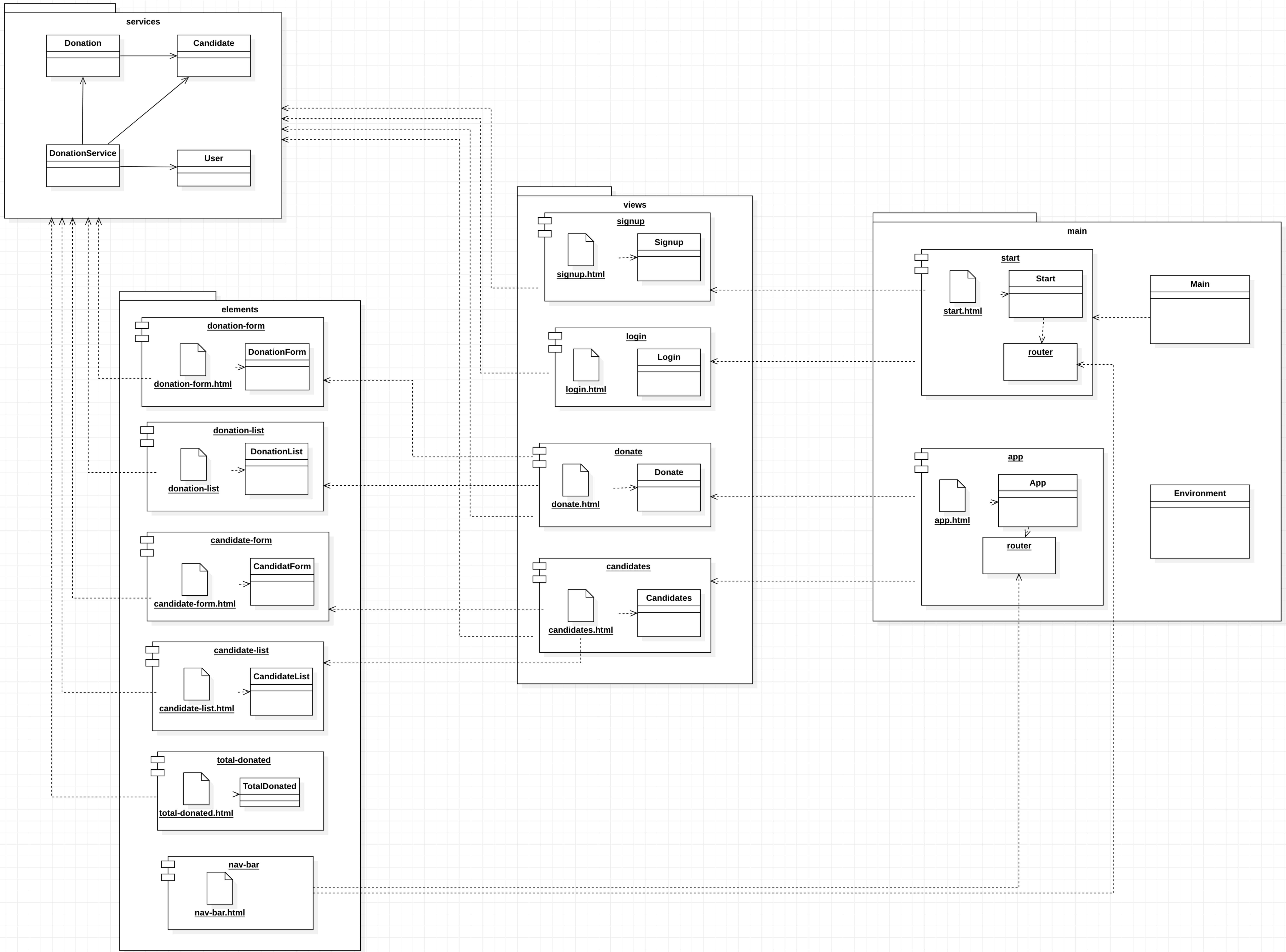
donate method



```
export interface Candidate {
  firstName: string;
  lastName: string;
  office: string;
}

export interface Donation {
  amount: number;
  method: string;
  candidate: Candidate;
}

export interface User {
  firstName: string;
  lastName: string;
  email: string;
  password: string;
}
```

- Donate method still local
- Does not attempt to contact donation-service

```
export class DonationService {


  async donate(amount: number, method: string, candidate: Candidate) {
    const donation = {
      amount: amount,
      method: method,
      candidate: candidate
    };
    this.donations.push(donation);
    this.total = this.total + amount;
    this.ea.publish(new TotalUpdate(this.total));
  }

}
```


services
Donation
Candidate
DonationService
User
elements
donation-form
donation-form.html
DonationForm
donation-list
donation-list
DonationList
candidate-form
candidate-form.html
CandidatForm
candidate-list
candidate-list.html
CandidateList
total-donated
total-donated.html
TotalDonated
nav-bar
nav-bar.html
views
signup
signup.html
Signup
login
login.html
Login
donate
donate.html
Donate
candidates
candidates.html
Candidates
main
start
start.html
Start
router
Main
app
app.html
App
router
Environment