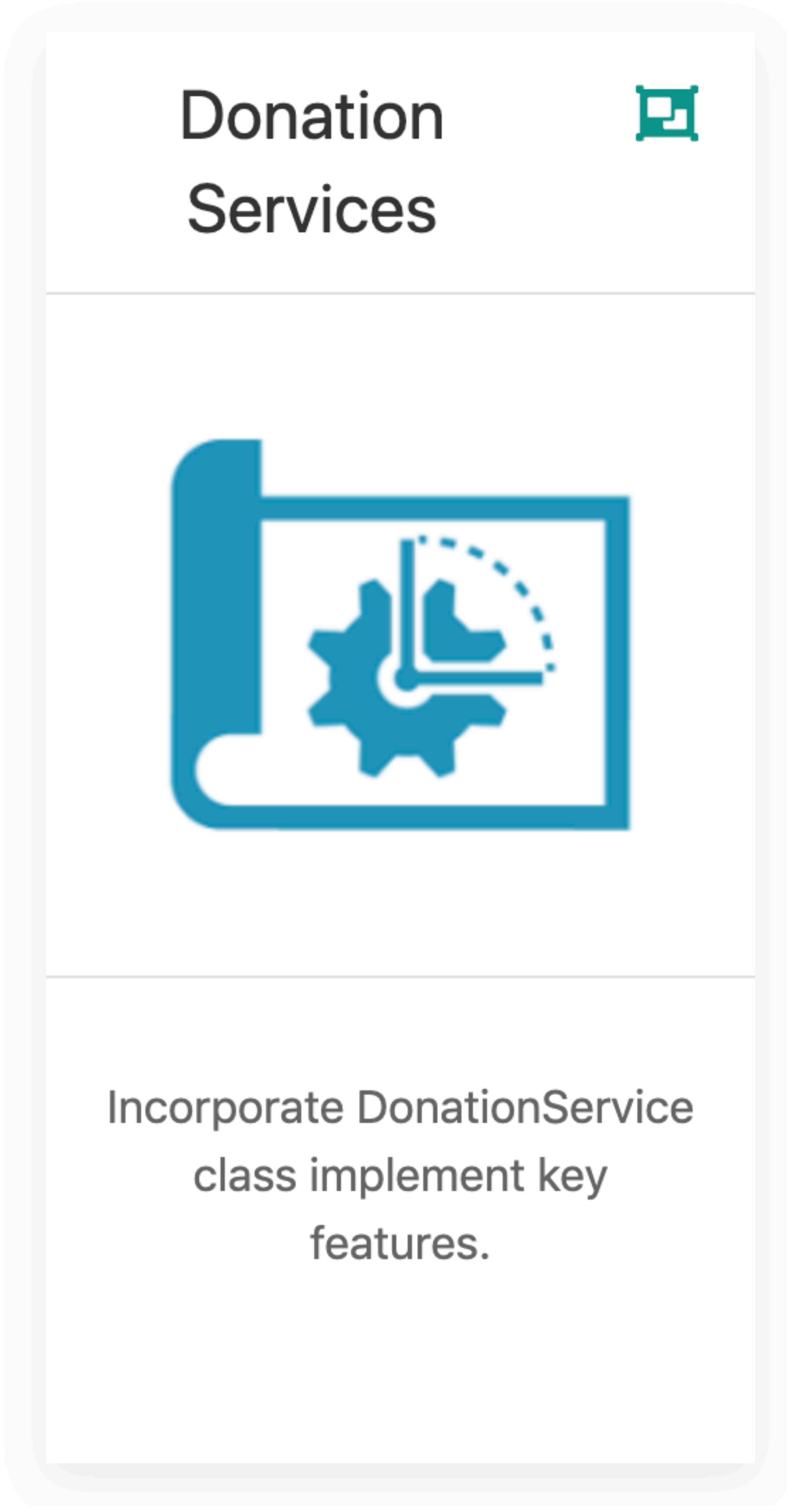
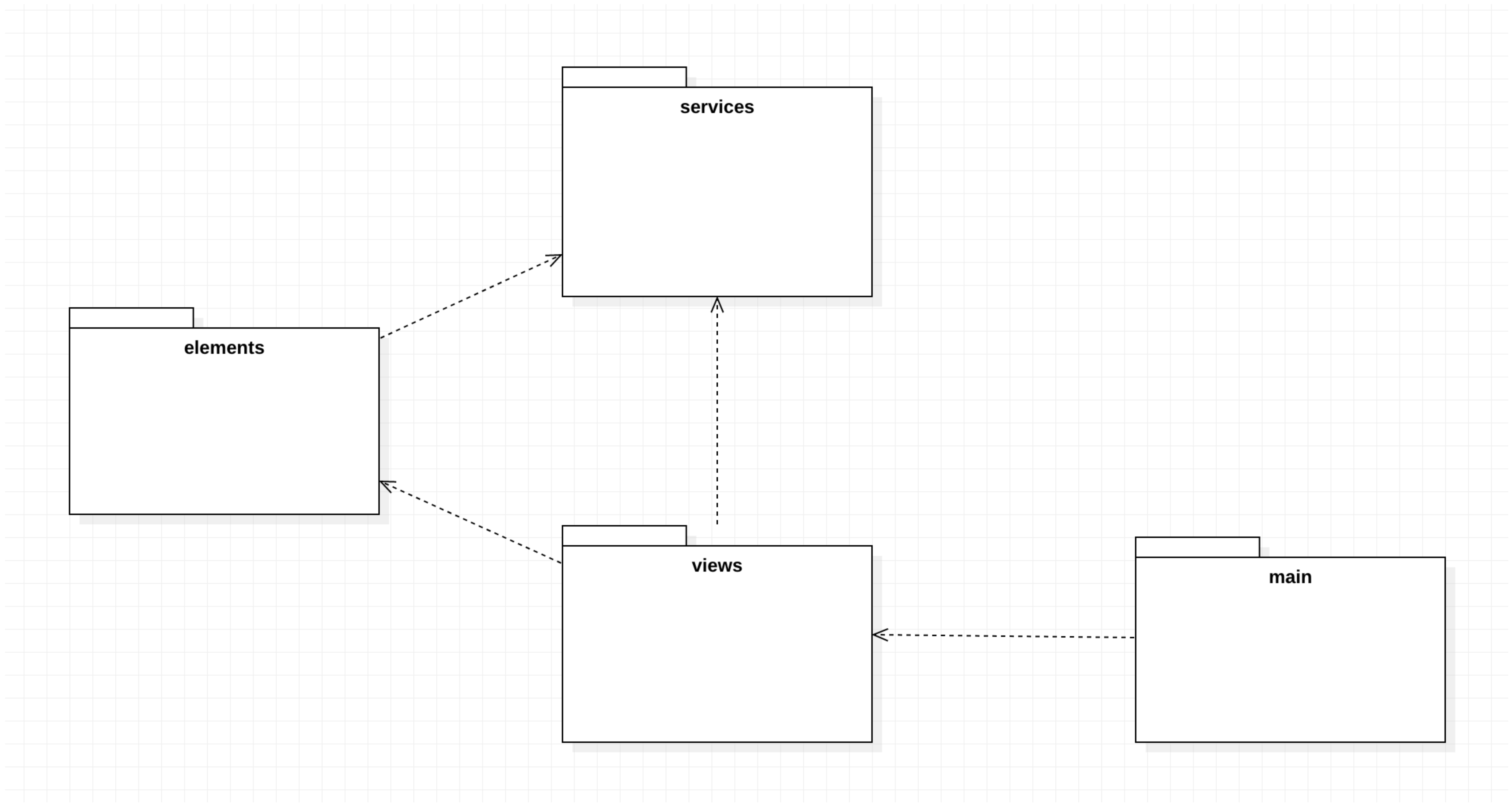
DonationService



Incorporate DonationService class implement key features.

Donation Client Architecture

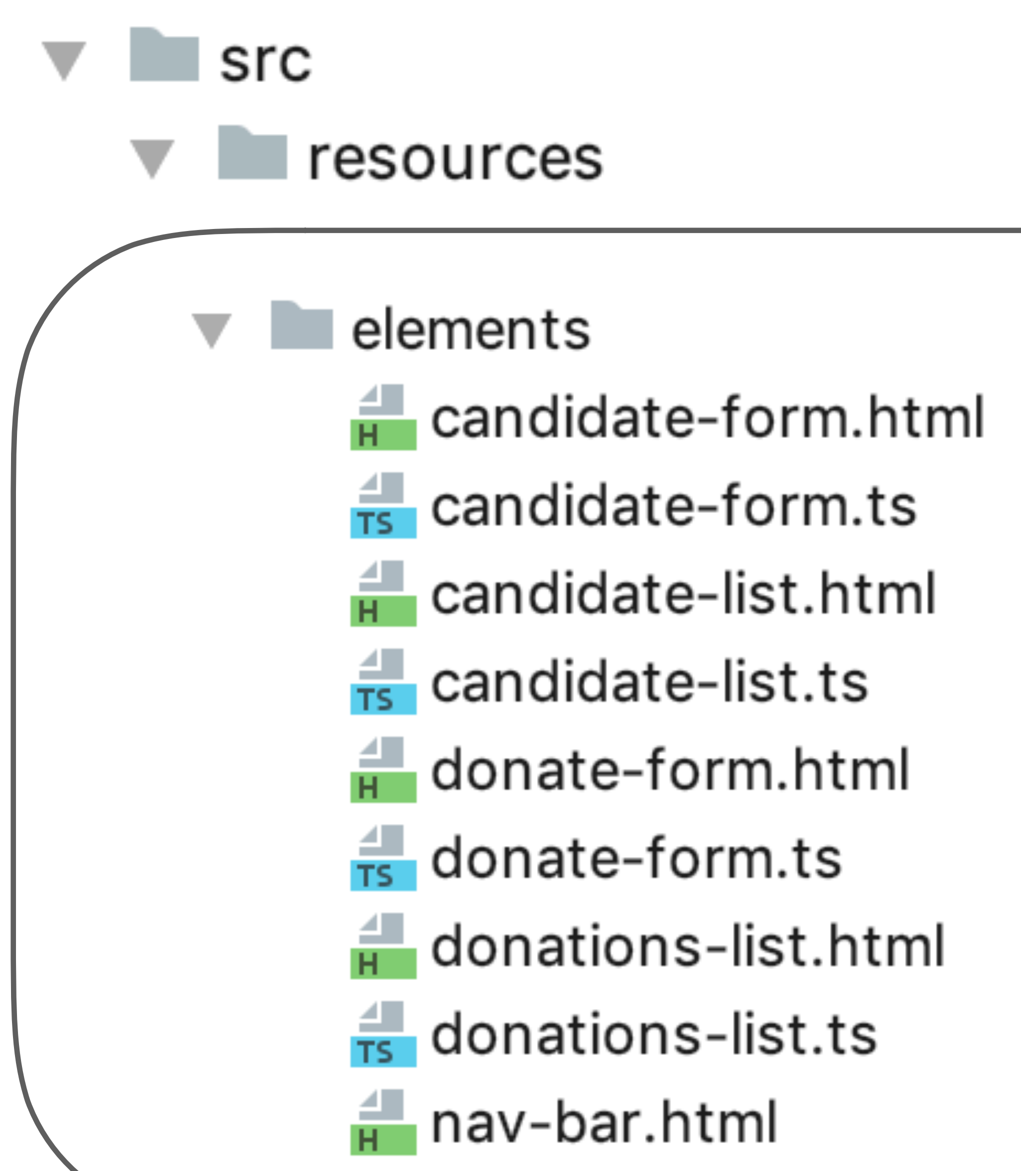


Application Components

```
▼ 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	candidate-form candidate-list nav-bar	donate-form donate-list	Elements
▼ services donation-types.ts	Candidate	Donation	Services
▼ views candidates.html candidates.ts donate.html donate.ts	candidates	donate	Views
app.html app.ts	app		Main

Application Components



```
▼ 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
```

candidate-form

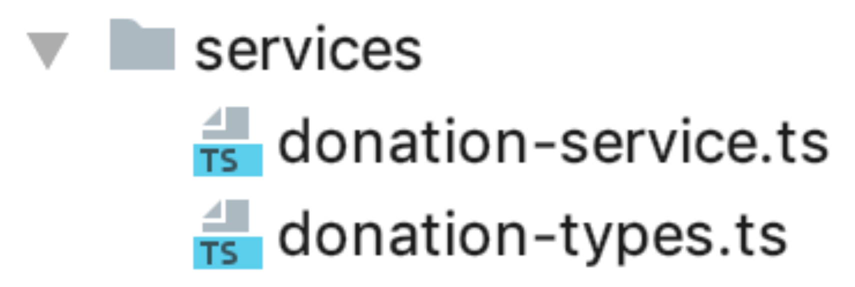
donate-form

candidate-list

donate-list

nav-bar

Elements



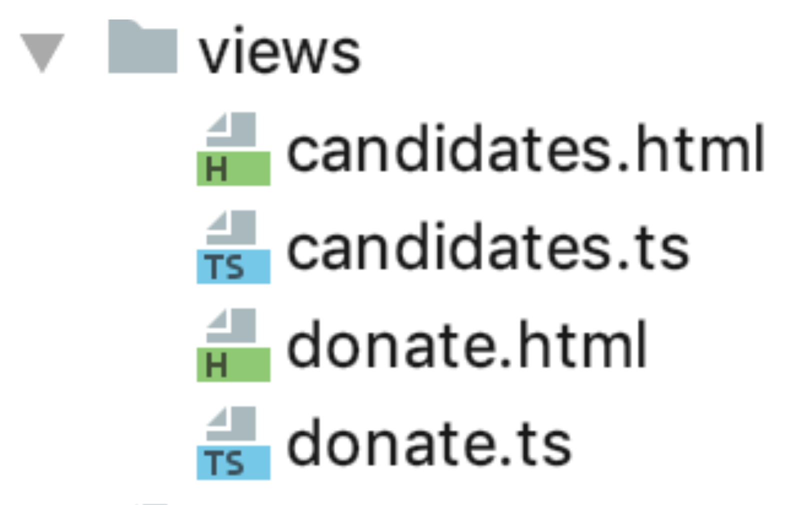
```
    ▼ services
        donation-service.ts
        donation-types.ts
```

Candidate

Donation

DonationService

Services

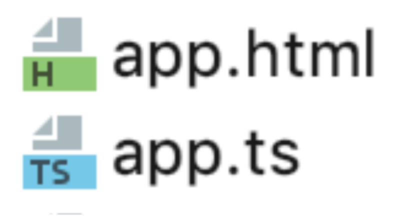


```
    ▼ views
        candidates.html
        candidates.ts
        donate.html
        donate.ts
```

candidates

donate

Views

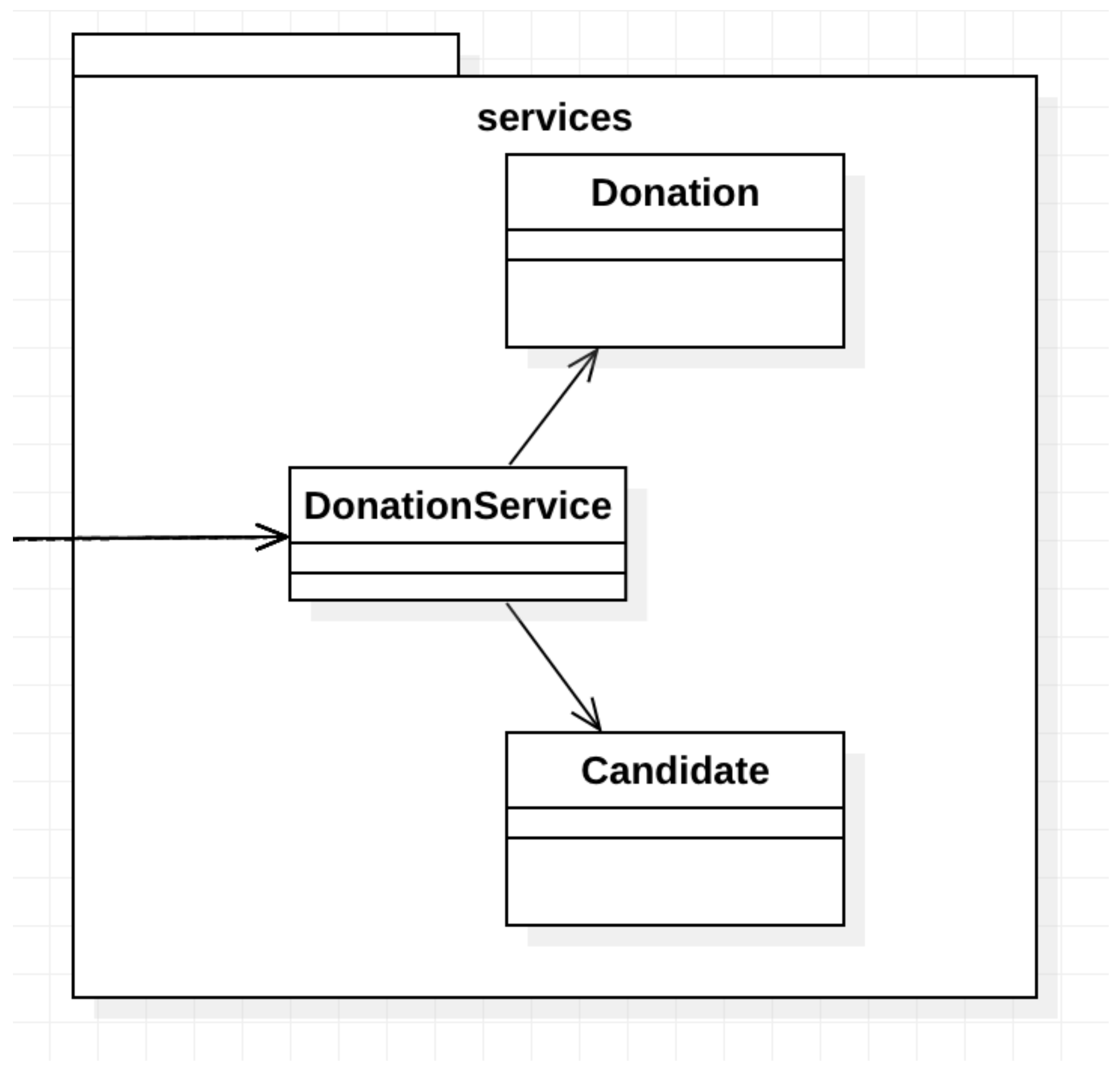


```
      app.html
      app.ts
```

app

Main

Services



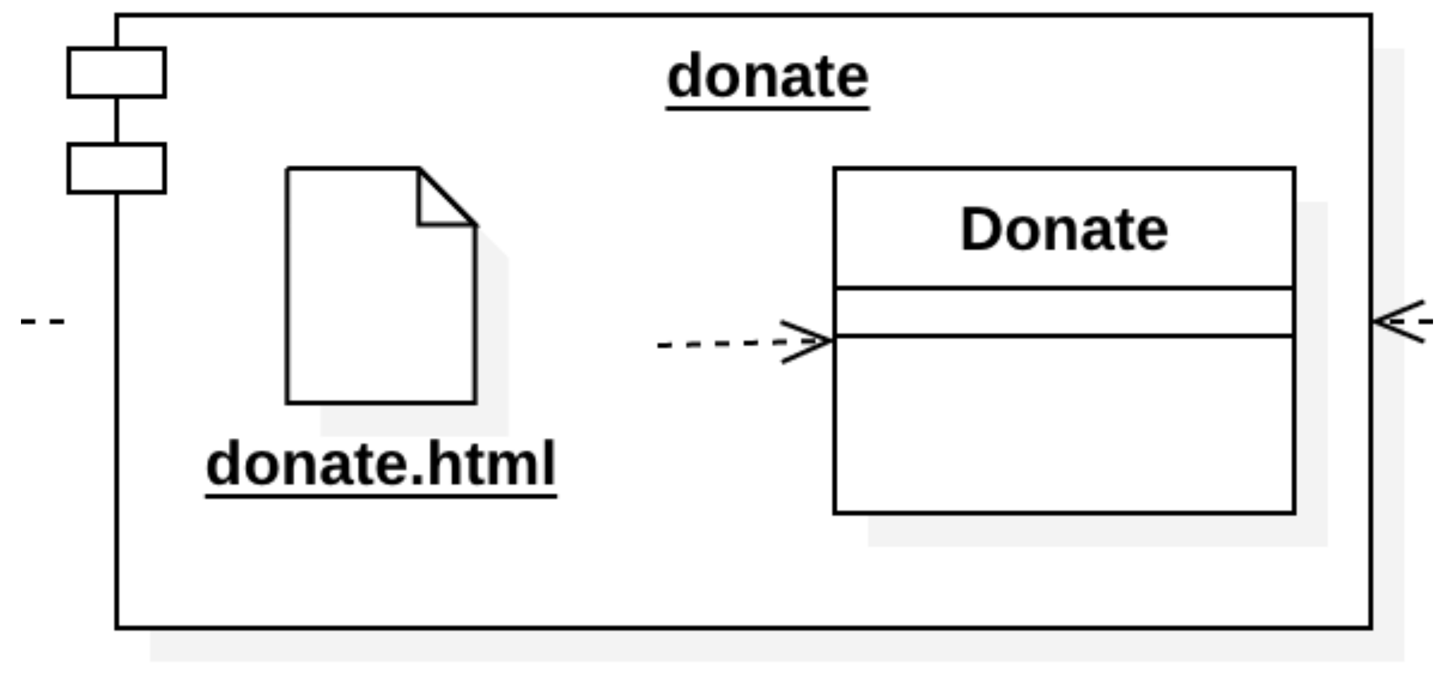
donation-types.ts

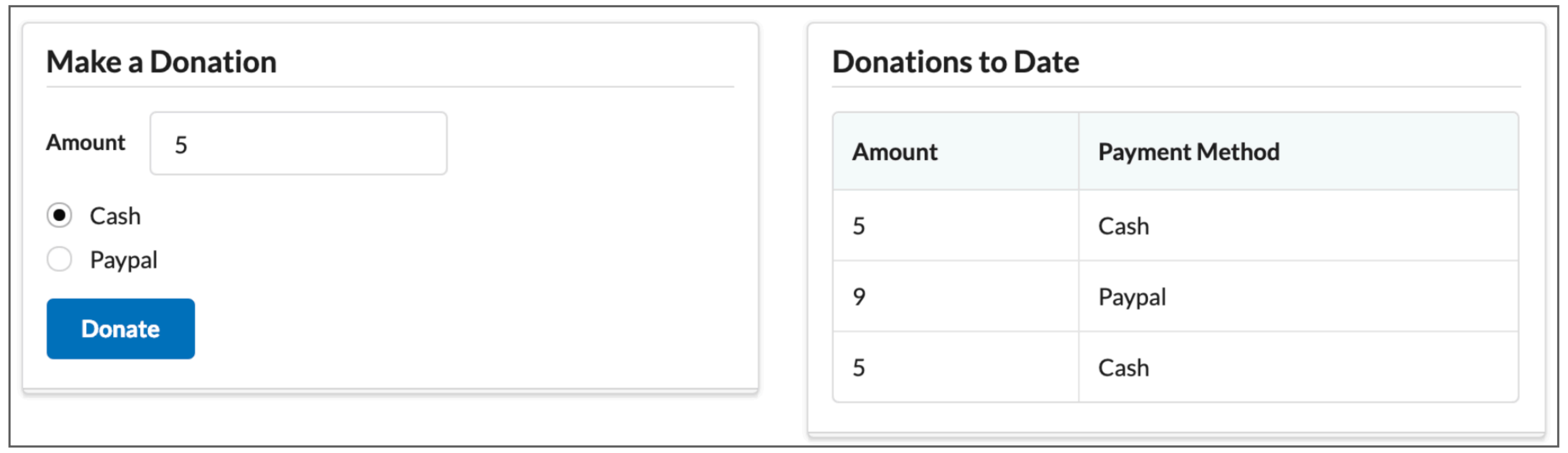
```
export interface Candidate {
  firstName: string;
  lastName: string;
  office: string;
}

export interface Donation {
  amount: number;
  method: string;
}
```

donation-service.ts

```
import { Candidate, Donation } from './donation-types';

export class DonationService {

  candidates: Candidate[] = [];
  donations: Donation[] = [];
  paymentMethods = ['Cash', 'Paypal'];
  total = 0;

  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;
    console.log('Total so far ' + this.total);
  }
}
```

Donate View Component





Make a Donation

Amount	5

● Cash
○ Paypal

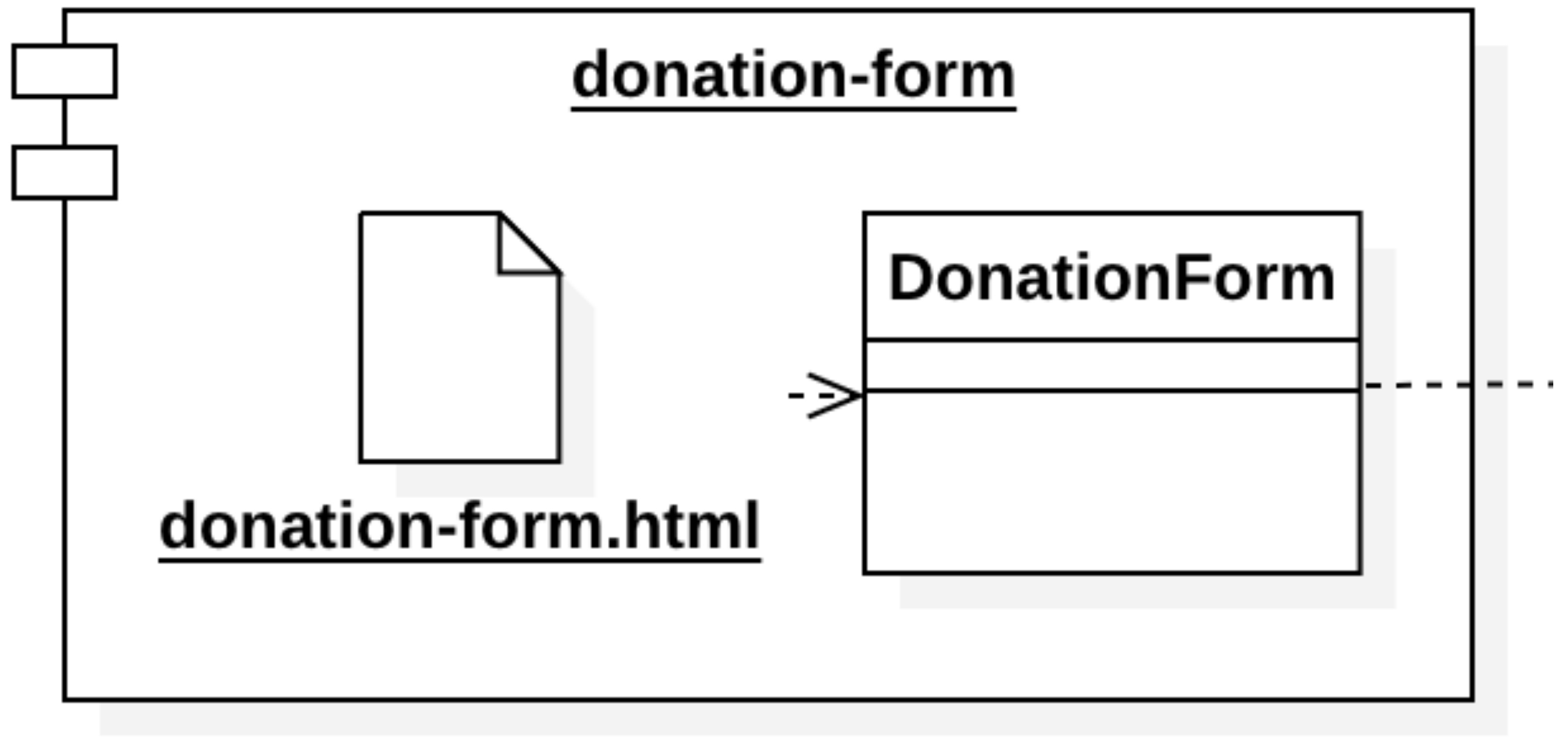
Donations to Date

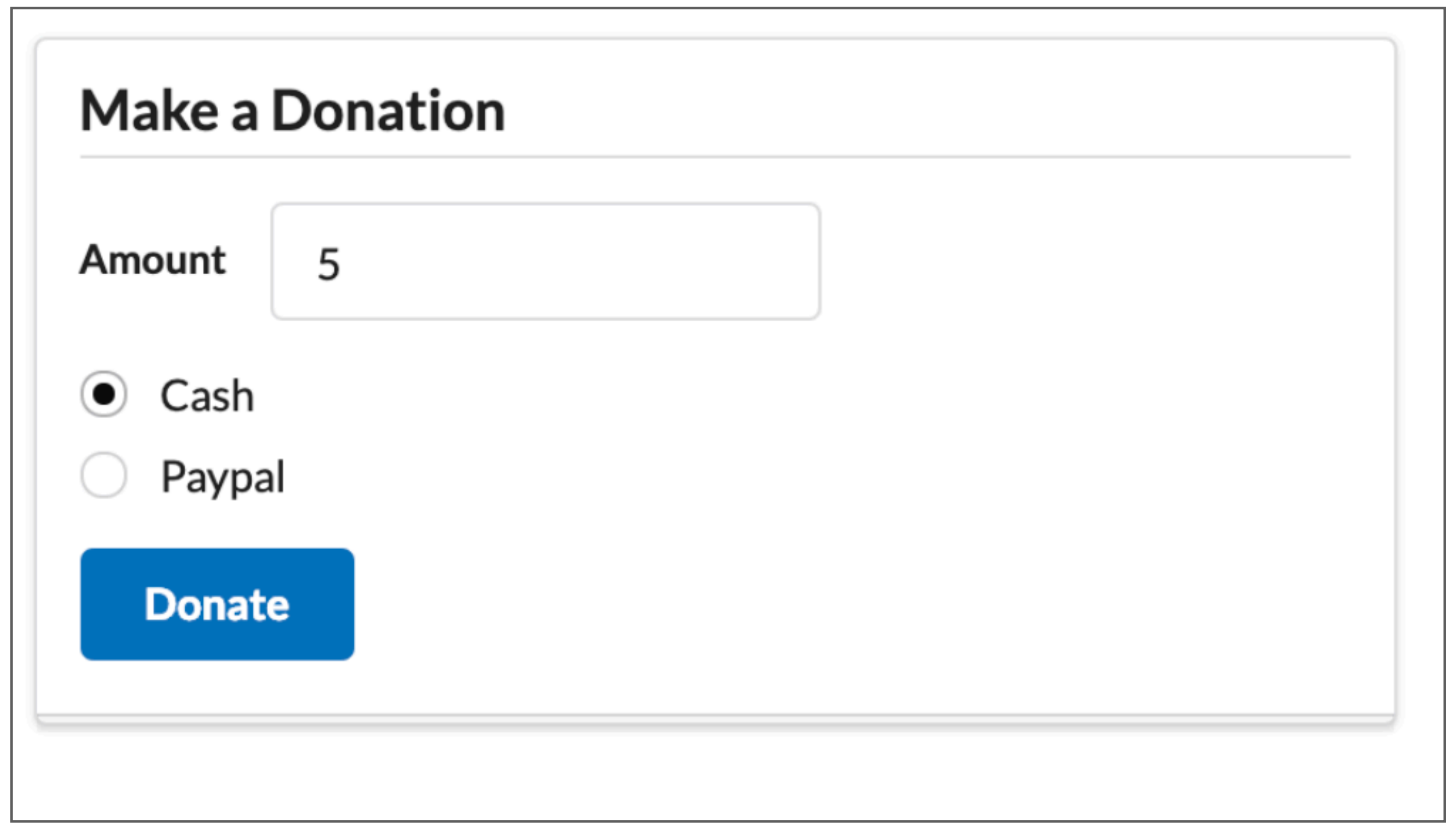
Amount	Payment Method
5	Cash
9	Paypal
5	Cash

```
import { inject } from 'aurelia-framework';
import { Candidate, Donation } from '../services/donation-types';
import { DonationService } from '../services/donation-service';

@inject(DonationService)
export class Donate {
  donations: Donation[];
  paymentMethods: string[];
  candidates: Candidate[];

  constructor(private ds: DonationService) {
    this.candidates = ds.candidates;
    this.donations = ds.donations;
    this.paymentMethods = ds.paymentMethods;
  }
}
```

```
<template>
  <require from="../resources/elements/donations-list"></require>
  <require from="../resources/elements/donate-form"></require>

  <div class="ui stackable two column grid">
    <div class="column">
      <donate-form donations.bind="donations" payment-methods.bind="paymentMethods" candidates.bind="candidates"></donate-form>
    </div>
    <div class="column">
      <donations-list donations.bind="donations"></donations-list>
    </div>
  </div>
</template>
```

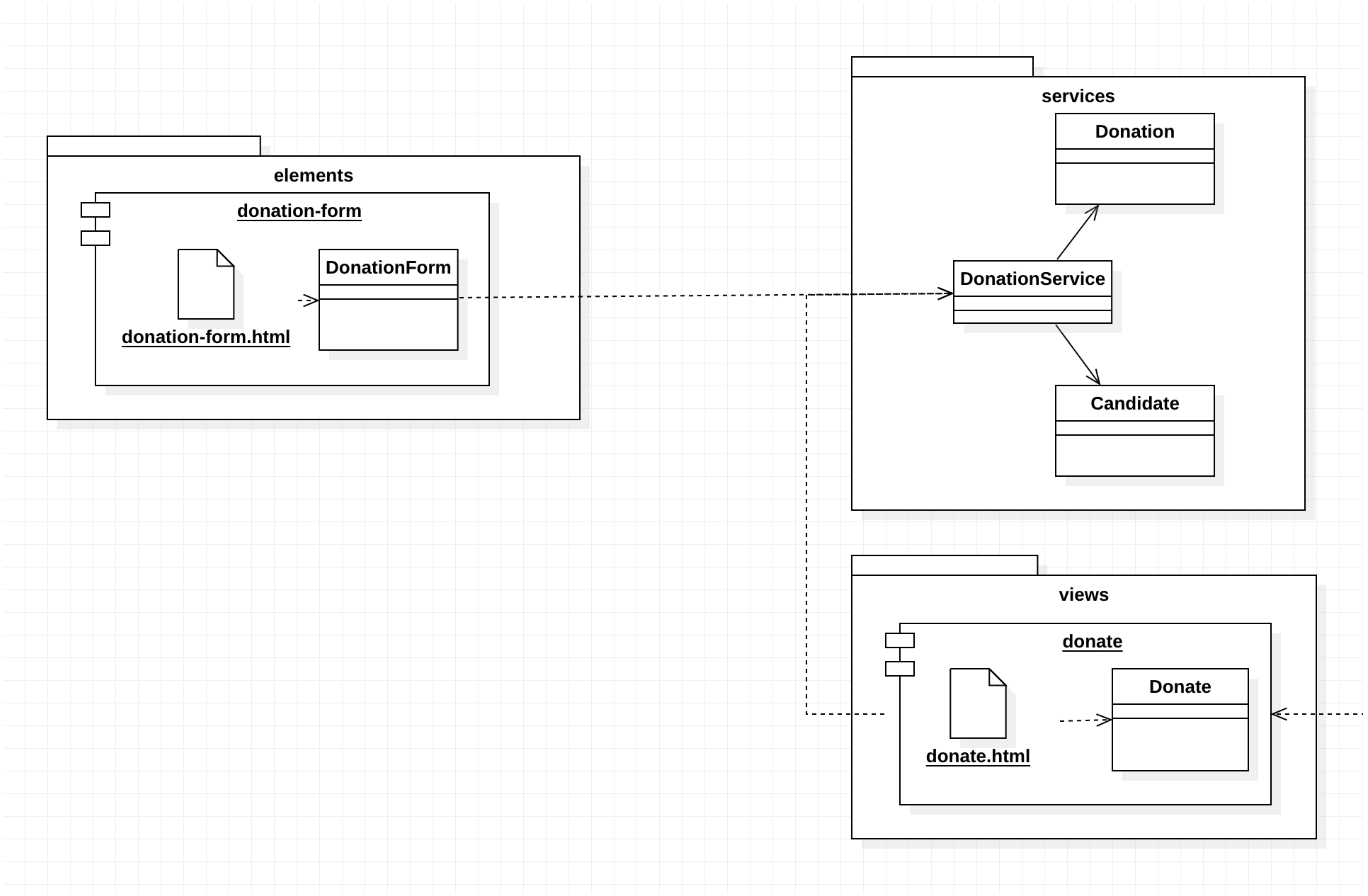
DonateFrom Custom Element Component

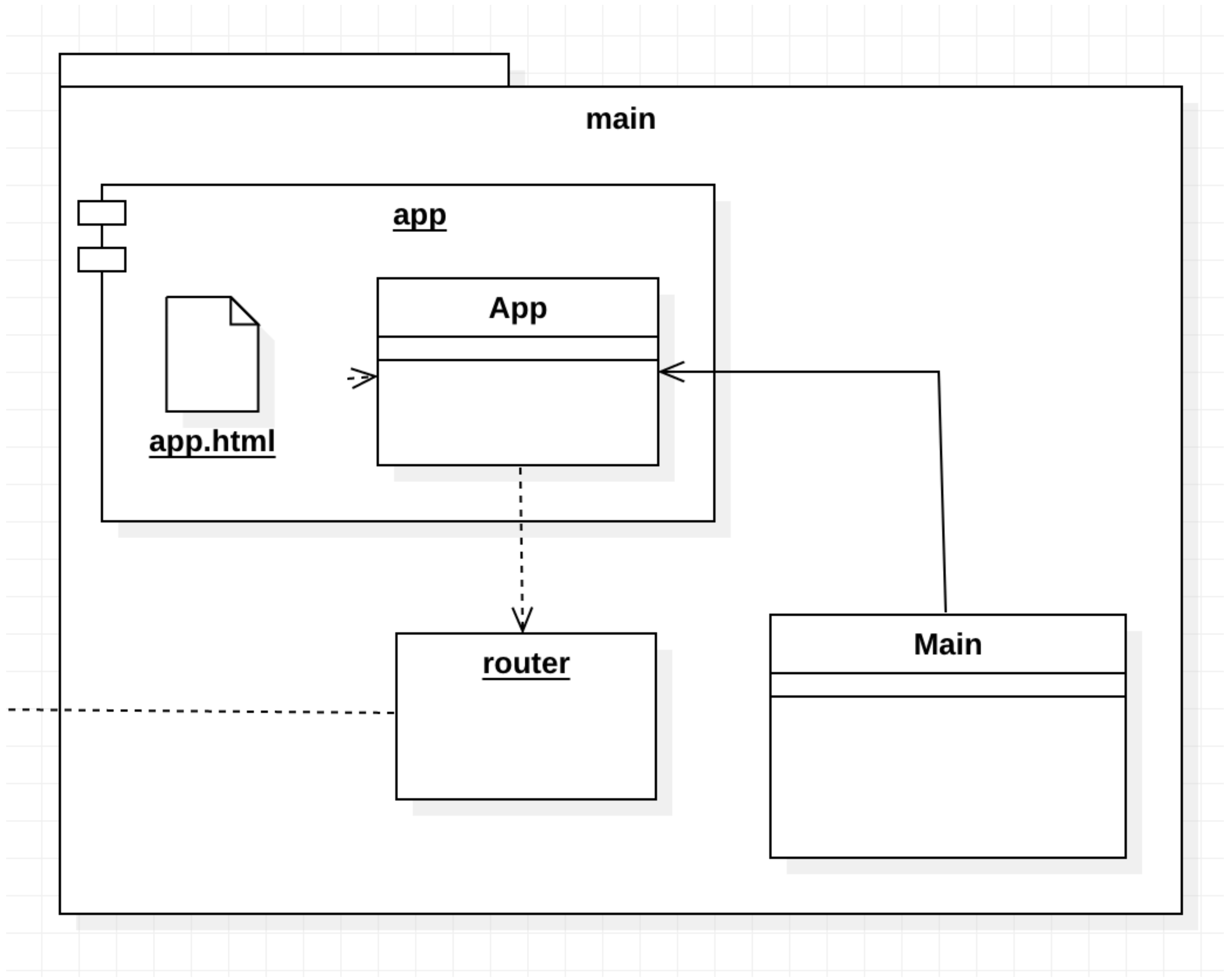




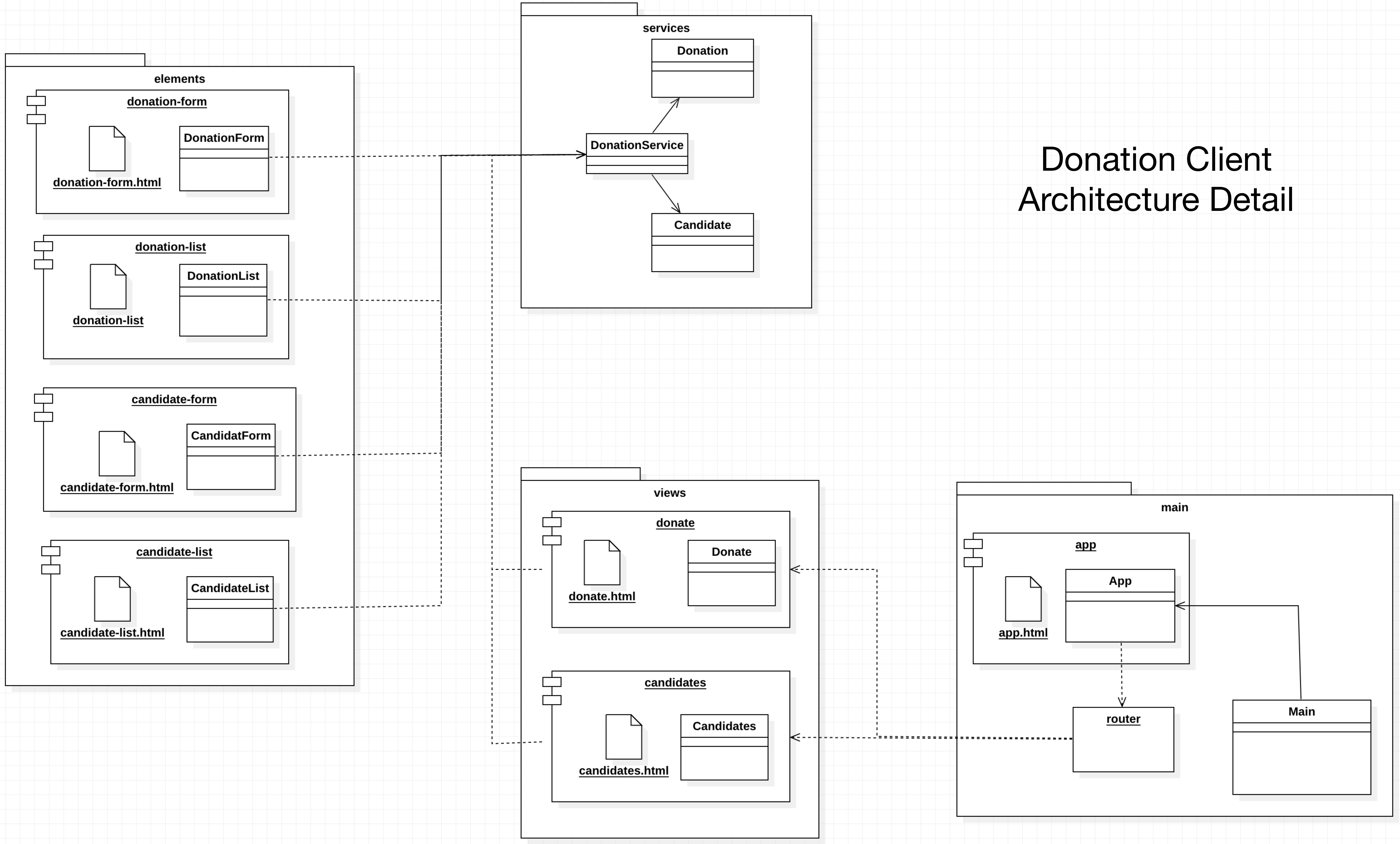
```
import { inject } from 'aurelia-framework';
import { bindable } from 'aurelia-framework';
import { Candidate, Donation } from '../../services/donation-types';
import {DonationService} from "../../services/donation-service";

@inject(DonationService)
export class DonateForm {
  @bindable
  paymentMethods: string[];
  @bindable
  candidates: Candidate[];

  amount = '0';
  selectedMethod = '';
  selectedCandidate : Candidate = null;

  constructor (private ds: DonationService) {}

  makeDonation() {
    this.ds.donate(parseInt(this.amount), this.selectedMethod, this.selectedCandidate);
  }
}
```

```
<template>
  <form submit.trigger="makeDonation()" class="ui form stacked segment">
    <h3 class='ui dividing header'> Make a Donation </h3>
    <div class="grouped inline fields">
      <div class="field">
        <label>Amount</label> <input type="number" value.bind="amount">
      </div>
    </div>
    <div class="grouped inline fields">
      <div class="field" repeat.for="method of paymentMethods">
        <div class="ui radio checkbox">
          <input type="radio" model.bind="method" checked.bind="selectedMethod">
          <label>${method}</label>
        </div>
      </div>
    </div>
    <h4 class="ui dividing header"> Select Candidate </h4>
    <div class="grouped inline fields">
      <div class="field" repeat.for="candidate of candidates">
        <div class="ui radio checkbox">
          <input type="radio" model.bind="candidate" checked.bind="selectedCandidate">
          <label>${candidate.lastName}, ${candidate.firstName}</label>
        </div>
      </div>
    </div>
    <button class="ui blue submit button">Donate</button>
  </form>
</template>
```


elements
donation-form
donation-form.html
DonationForm
services
Donation
DonationService
Candidate
views
donate
donate.html
Donate


main
app
App
app.html
router
Main

Donation Client Architecture Detail


elements
donation-form
donation-form.html
DonationForm
donation-list
donation-list
DonationList
candidate-form
candidate-form.html
CandidatForm
candidate-list
candidate-list.html
CandidateList
services
Donation
DonationService
Candidate
views
donate
donate.html
Donate
candidates
candidates.html
Candidates
main
app
app.html
App
router
Main