Extending the API

Donation API

```
module.exports = [
  { method: 'GET', path: '/api/candidates', config: CandidatesApi.find },
  { method: 'GET', path: '/api/candidates/{id}', config: CandidatesApi.findOne },
  { method: 'POST', path: '/api/candidates', config: CandidatesApi.create },
  { method: 'DELETE', path: '/api/candidates/{id}', config: CandidatesApi.deleteOne },
  { method: 'DELETE', path: '/api/candidates', config: CandidatesApi.deleteAll },

  { method: 'GET', path: '/api/users', config: UsersApi.find },
  { method: 'GET', path: '/api/users/{id}', config: UsersApi.findOne },
  { method: 'POST', path: '/api/users', config: UsersApi.create },
  { method: 'DELETE', path: '/api/users/{id}', config: UsersApi.deleteOne },
  { method: 'DELETE', path: '/api/users', config: UsersApi.deleteAll },
];
```

- Supports creation/deletion of Users & Candidates
- Comprehensive set of unit tests in place
- Solid foundation for extending the API

Donations

- Create, Delete and Retrieve donations needs to be supported
- Retrieve Donations options:
    - All donations recorded for all Candidates
    - All Donations for a single candidate
    - All donations made by an singe donor
- Create Donation
    - By a Donor to a Candidate

Retrieve All Donations

routesapi.js

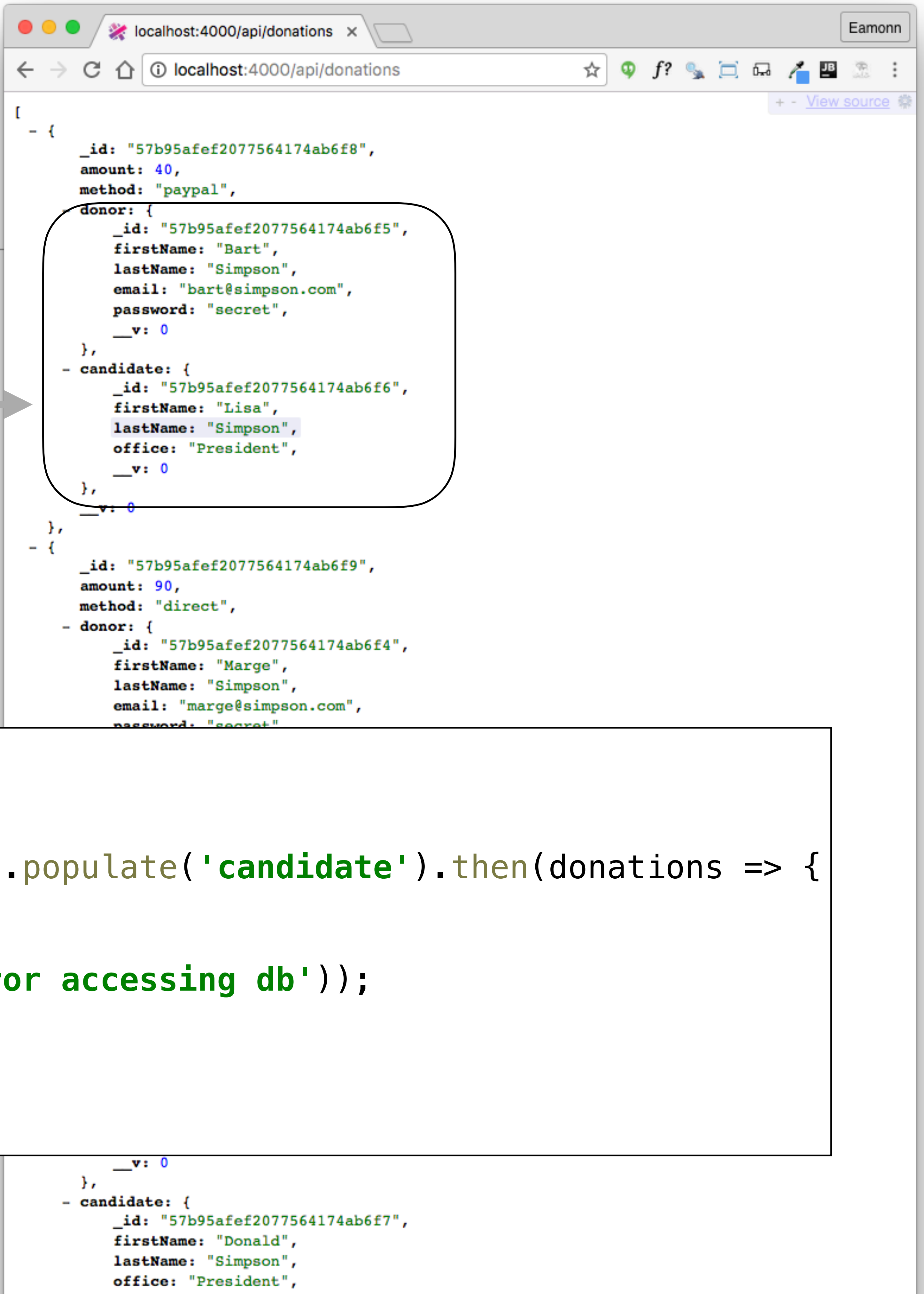
```
...
const DonationsApi = require('./app/api/donationsapi');
...

  { method: 'GET', path: '/api/donations', config: CandidatesApi.findAllDonations },
...
```

api/donationsapi.js

```
'use strict';

const Donation = require('../models/donation');
const Boom = require('boom');

exports.findAllDonations = {

  auth: false,

  handler: function (request, reply) {
    Donation.find({}).populate('donor').populate('candidate').then(donations => {
      reply(donations);
    }).catch(err => {
      reply(Boom.badImplementation('error accessing db'));
    });
  },
};
```

populate

- Populate ensures object references expanded in json



```
[
  - {
        _id: "57b95afef2077564174ab6f8",
        amount: 40,
        method: "paypal",
      - donor: {
            _id: "57b95afef2077564174ab6f5",
            firstName: "Bart",
            lastName: "Simpson",
            email: "bart@simpson.com",
            password: "secret",
            __v: 0
        },
      - candidate: {
            _id: "57b95afef2077564174ab6f6",
            firstName: "Lisa",
            lastName: "Simpson",
            office: "President",
            __v: 0
        },
        __v: 0
    },
  - {
        _id: "57b95afef2077564174ab6f9",
        amount: 90,
        method: "direct",
      - donor: {
            _id: "57b95afef2077564174ab6f4",
            firstName: "Marge",
            lastName: "Simpson",
            email: "marge@simpson.com",
            password: "secret"
```

```
            __v: 0
        },
      - candidate: {
            _id: "57b95afef2077564174ab6f7",
            firstName: "Donald",
            lastName: "Simpson",
            office: "President",
```

```
exports.findAllDonations = {

  handler: function (request, reply) {
    Donation.find({}).populate('donor').populate('candidate').then(donations => {
      reply(donations);
    }).catch(err => {
      reply(Boom.badImplementation('error accessing db'));
    });
  },
};
```

Retrieve Donations for a specific Candidate

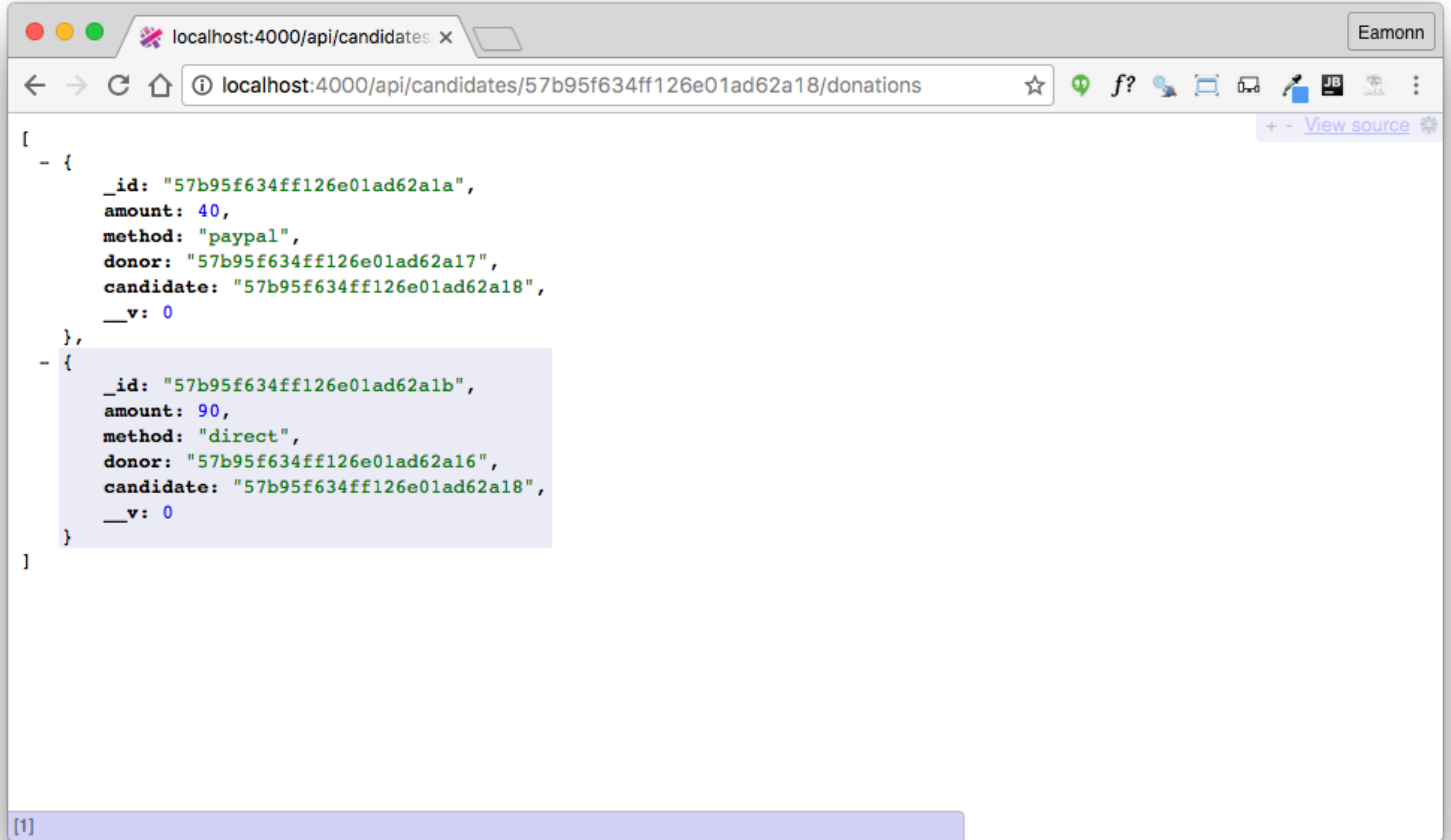
routesapi.js

```
  { method: 'GET', path: '/api/candidates/{id}/donations', config: DonationsApi.findDonations },
```

donationsapi.js

```
exports.findDonations = {

  auth: false,

  handler: function (request, reply) {
    Donation.find({ candidate: request.params.id }).then(donations => {
      reply(donations);
    }).catch(err => {
      reply(Boom.badImplementation('error accessing db'));
    });
  },

};
```

http://localhost:4000/api/candidates/$\overbrace{\text{012345678901234567890123}}^{\text{candidate id}}$/donations



localhost:4000/api/candidates/57b95f634ff126e01ad62a18/donations

```
[
  - {
        _id: "57b95f634ff126e01ad62a1a",
        amount: 40,
        method: "paypal",
        donor: "57b95f634ff126e01ad62a17",
        candidate: "57b95f634ff126e01ad62a18",
        __v: 0
    },
  - {
        _id: "57b95f634ff126e01ad62a1b",
        amount: 90,
        method: "direct",
        donor: "57b95f634ff126e01ad62a16",
        candidate: "57b95f634ff126e01ad62a18",
        __v: 0
    }
]
```

Create a Donation

routesapi.js

```
{ method: 'POST', path: '/api/candidates/{id}/donations', config: CandidatesApi.makeDonation },
```

app/api/donationsapi.js

```
exports.makeDonation = {

  auth: false,

  handler: function (request, reply) {
    const donation = new Donation(request.payload);
    donation.candidate = request.params.id;
    donation.save().then(newDonation => {
      reply(newDonation).code(201);
    }).catch(err => {
      reply(Boom.badImplementation('error making donation'));
    });
  },

};
```

Delete all Donations

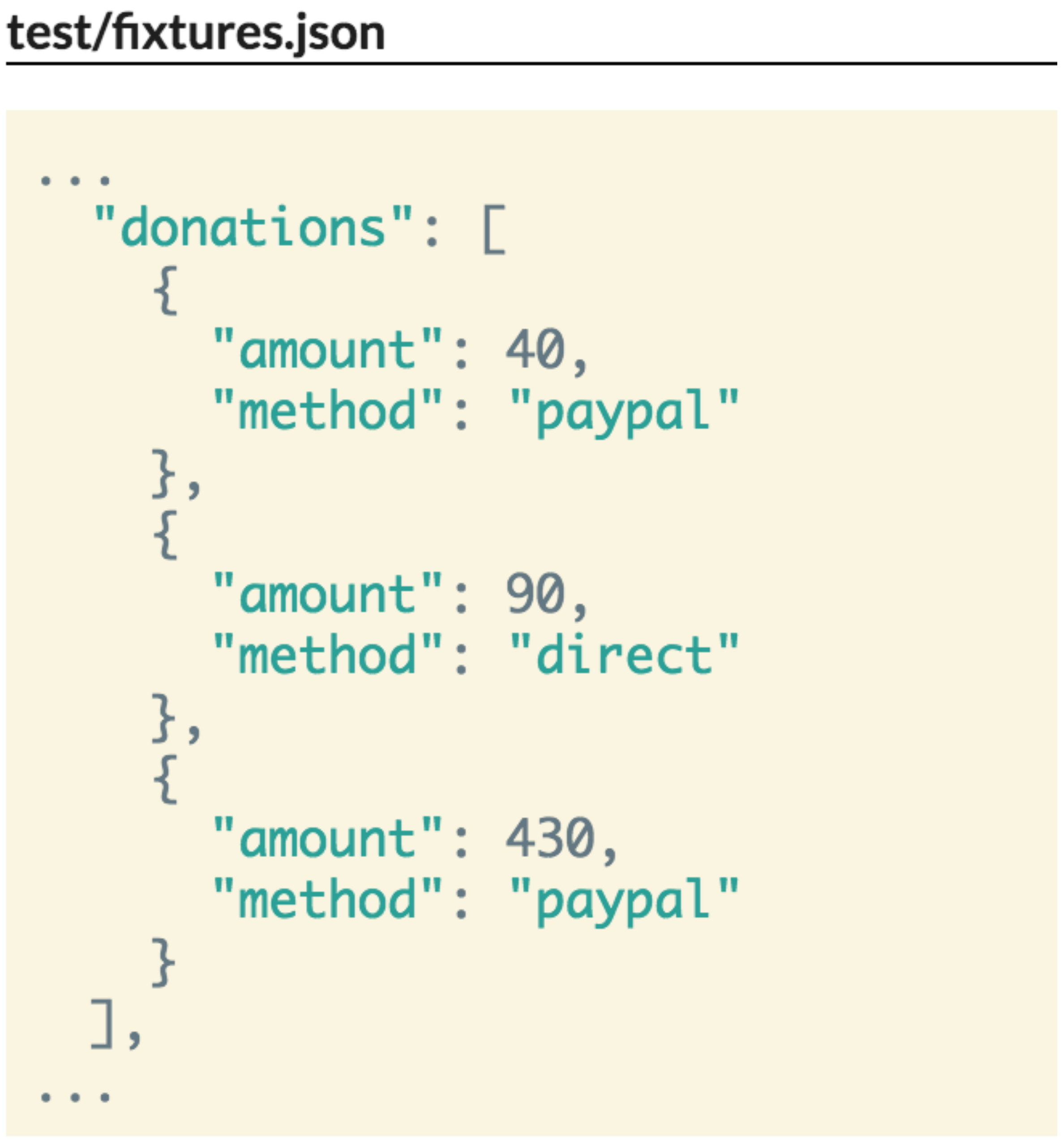
routesapi.js

```
{ method: 'DELETE', path: '/api/donations', config: DonationsApi.deleteAllDonations },
```

```
exports.deleteAllDonations = {

  auth: false,

  handler: function (request, reply) {
    Donation.remove({}).then(err => {
      reply().code(204);
    }).catch(err => {
      reply(Boom.badImplementation('error removing Donations'));
    });
  },

};
```

Testing the API - Fixtures

- Additional Fixture data



test/fixtures.json

```
...
  "donations": [
    {
      "amount": 40,
      "method": "paypal"
    },
    {
      "amount": 90,
      "method": "direct"
    },
    {
      "amount": 430,
      "method": "paypal"
    }
  ],
...
```

Testing the API - DonationService Support

donation-service.js

```
makeDonation(id, donation) {
  return this.httpService.post('/api/candidates/' + id + '/donations', donation);
}

getDonations(id) {
  return this.httpService.get('/api/candidates/' + id + '/donations');
}

deleteAllDonations() {
  return this.httpService.delete('/api/donations');
}
```

- Additional methods in DonationService class

Test Skeleton

test/donationsapitest.js

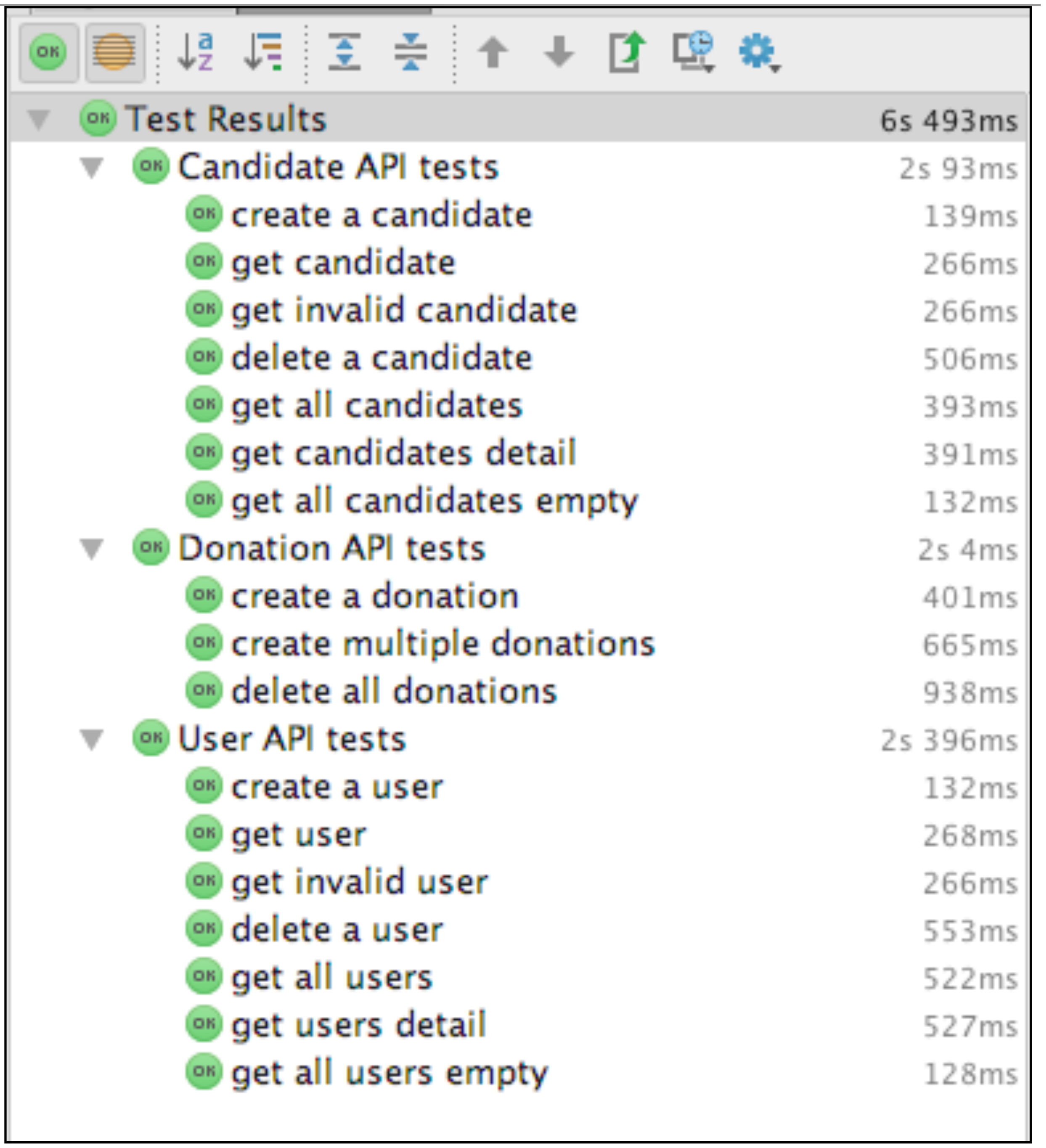
```
'use strict';

const assert = require('chai').assert;
const DonationService = require('./donation-service');
const fixtures = require('./fixtures.json');
const _ = require('lodash');

suite('Donation API tests', function () {

  let donations = fixtures.donations;
  let newCandidate = fixtures.newCandidate;

  const donationService = new DonationService(fixtures.donationService);

  beforeEach(function () {
  });

  afterEach(function () {

  });

  test('a test function', function () {

  });

});
```

- Comprehensive test of the Donations endpoints

```
'use strict';

const assert = require('chai').assert;
const DonationService = require('./donation-service');
const fixtures = require('./fixtures.json');
const _ = require('lodash');

suite('Donation API tests', function () {

  let donations = fixtures.donations;
  let newCandidate = fixtures.newCandidate;

  const donationService = new DonationService(fixtures.donationService);

  beforeEach(function () {
    donationService.deleteAllCandidates();
    donationService.deleteAllDonations();
  });

  afterEach(function () {
    donationService.deleteAllCandidates();
    donationService.deleteAllDonations();
  });

  test('create a donation', function () {
    const returnedCandidate = donationService.createCandidate(newCandidate);
    donationService.makeDonation(returnedCandidate._id, donations[0]);
    const returnedDonations = donationService.getDonations(returnedCandidate._id);
    assert.equal(returnedDonations.length, 1);
    assert(_.some([returnedDonations[0]], donations[0]), 'returned donation must be a superset of donation');
  });

  test('create multiple donations', function () {
    const returnedCandidate = donationService.createCandidate(newCandidate);
    for (var i = 0; i < donations.length; i++) {
      donationService.makeDonation(returnedCandidate._id, donations[i]);
    }

    const returnedDonations = donationService.getDonations(returnedCandidate._id);
    assert.equal(returnedDonations.length, donations.length);
    for (var i = 0; i < donations.length; i++) {
      assert(_.some([returnedDonations[i]], donations[i]), 'returned donation must be a superset of donation');
    }
  });

  test('delete all donations', function () {
    const returnedCandidate = donationService.createCandidate(newCandidate);
    for (var i = 0; i < donations.length; i++) {
      donationService.makeDonation(returnedCandidate._id, donations[i]);
    }

    const d1 = donationService.getDonations(returnedCandidate._id);
    assert.equal(d1.length, donations.length);
    donationService.deleteAllDonations();
    const d2 = donationService.getDonations(returnedCandidate._id);
    assert.equal(d2.length, 0);
  });
});
```

Regression Tests

- Powerful capability to comprehensively test the API
- Further development/ enhancement of the API not has large range of regression tests
- These will keep the API development stable as new endpoints are introduced



```
Test Results                          6s 493ms
  Candidate API tests                 2s 93ms
    create a candidate                139ms
    get candidate                     266ms
    get invalid candidate             266ms
    delete a candidate                506ms
    get all candidates                393ms
    get candidates detail             391ms
    get all candidates empty          132ms
  Donation API tests                  2s 4ms
    create a donation                 401ms
    create multiple donations         665ms
    delete all donations              938ms
  User API tests                      2s 396ms
    create a user                     132ms
    get user                          268ms
    get invalid user                  266ms
    delete a user                     553ms
    get all users                     522ms
    get users detail                  527ms
    get all users empty               128ms
```