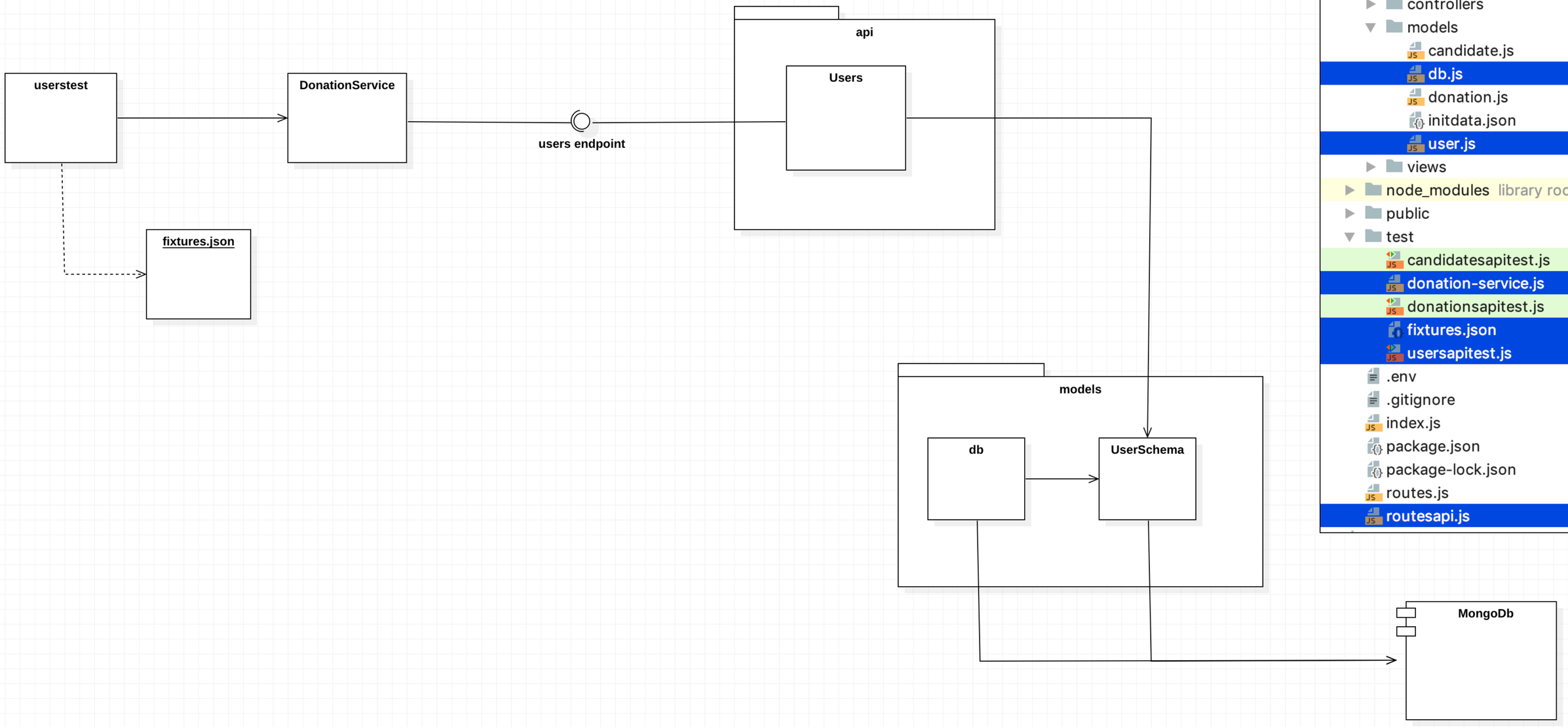
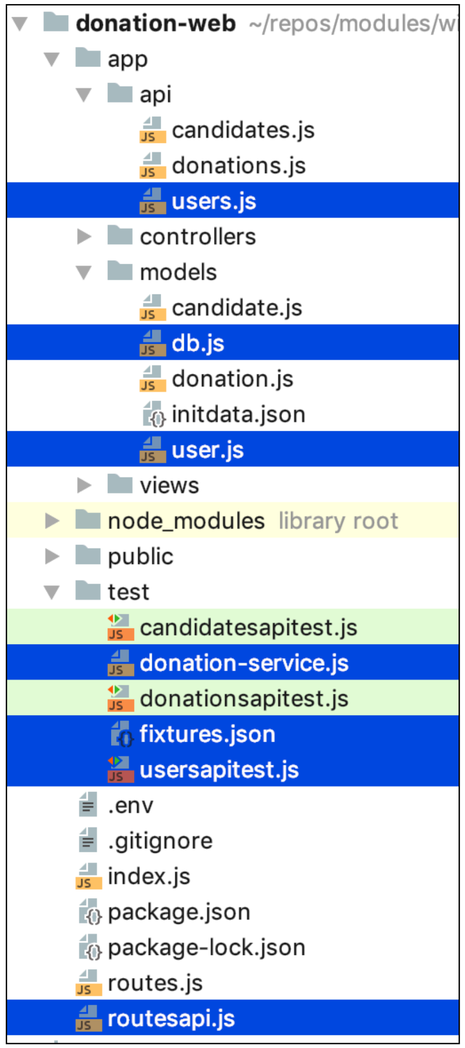
Users Endpoint


userstest
DonationService
users endpoint
api
Users
fixtures.json
models
db
UserSchema
MongoDb

donation-web ~/repos/modules/w
app
api
candidates.js
donations.js
users.js
controllers
models
candidate.js
db.js
donation.js
initdata.json
user.js
views
node_modules library root
public
test
candidatesapitest.js
donation-service.js
donationsapitest.js
fixtures.json
usersapitest.js
.env
.gitignore
index.js
package.json
package-lock.json
routes.js
routesapi.js

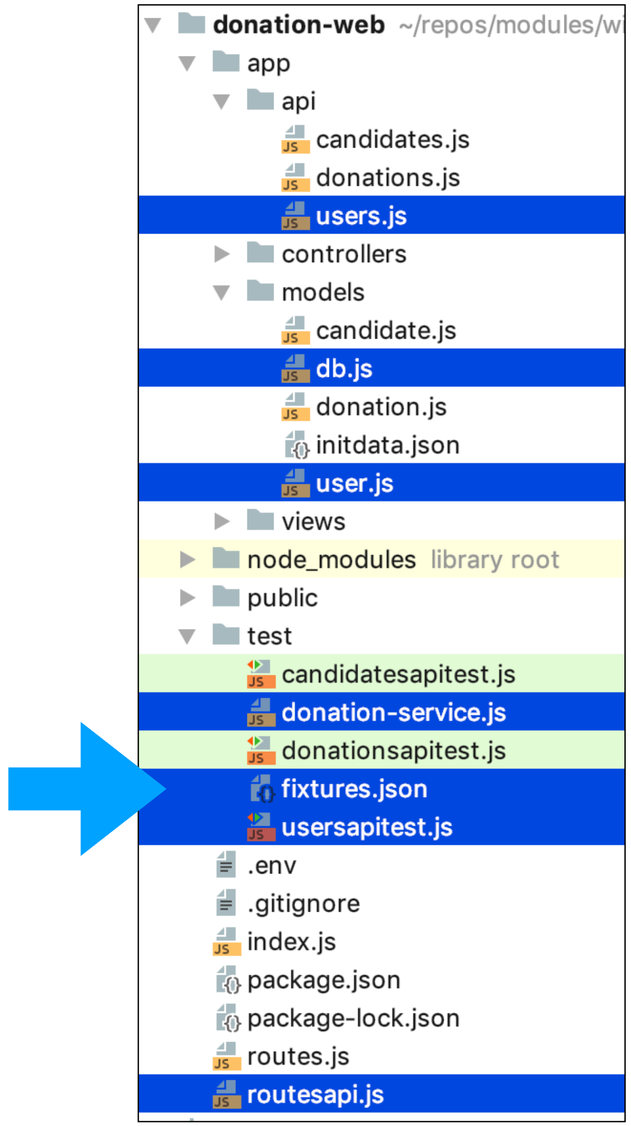
- Fixtures
- Routes
- Controller
- DonationService
- Unit Tests



Fixtures

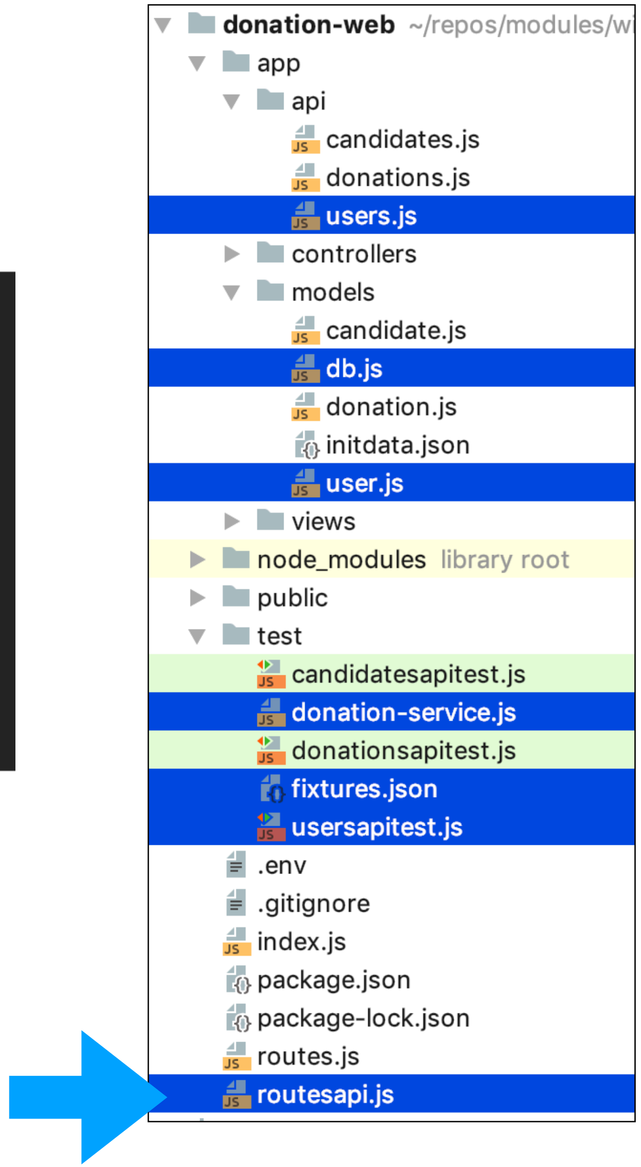
- Fixtures to enhance reusability & readability of unit tests

```
{
  "donationService": "http://localhost:3000",
  "users": [
    {
      "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"
    }
  ],
  "candidates": [
    {
      "firstName": "Lisa",
      "lastName": "Simpson",
      "office": "President"
    },
    {
      "firstName": "Donald",
      "lastName": "Simpson",
      "office": "President"
    }
  ],
  "newCandidate": {
    "firstName": "Barnie",
    "lastName": "Grumble",
    "office": "President"
  },
  "newUser": {
    "firstName": "Maggie",
    "lastName": "Simpson",
    "email": "maggie@simpson.com",
    "password": "secret"
  }
}
```



Routes

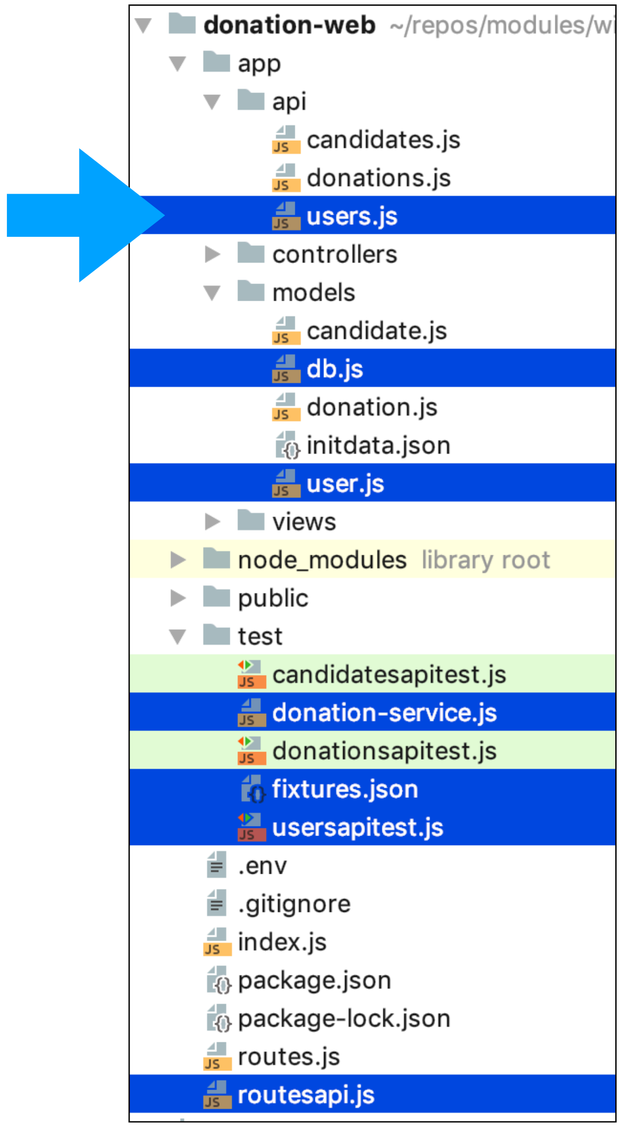
```
const Users= require('./app/api/users');

...

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



Controller

- Users Endpoint
  - find
  - findOne
  - create

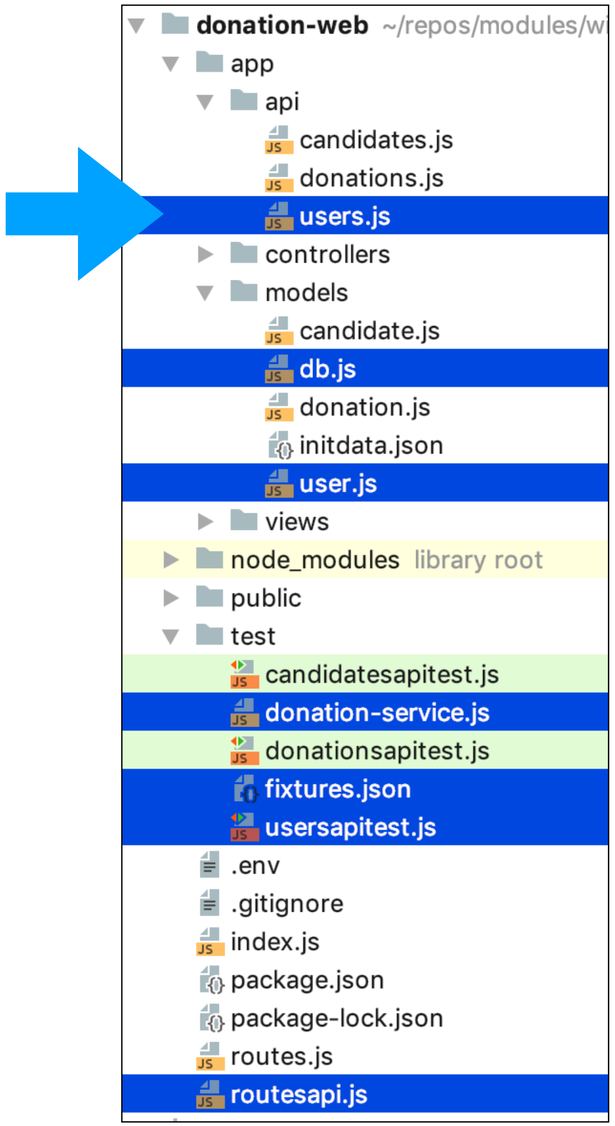
```
'use strict';

const Boom = require('boom');
const User = require('../models/user');

const Users = {

  find: {
    auth: false,
    handler: async function(request, h) {
      const users = await User.find();
      return users;
    }
  },

  findOne: {
    auth: false,
    handler: async function(request, h) {
      try {
        const user = await User.findOne({ _id: request.params.id });
        if (!user) {
          return Boom.notFound('No User with this id');
        }
        return user;
      } catch (err) {
        return Boom.notFound('No User with this id');
      }
    }
  },

  create: {
    auth: false,
    handler: async function(request, h) {
      const newUser = new User(request.payload);
      const user = await newUser.save();
      if (user) {
        return h.response(user).code(201);
      }
      return Boom.badImplementation('error creating user');
    }
  },
```



Controller

- Users Endpoint
  - deleteAll
  - deleteOne

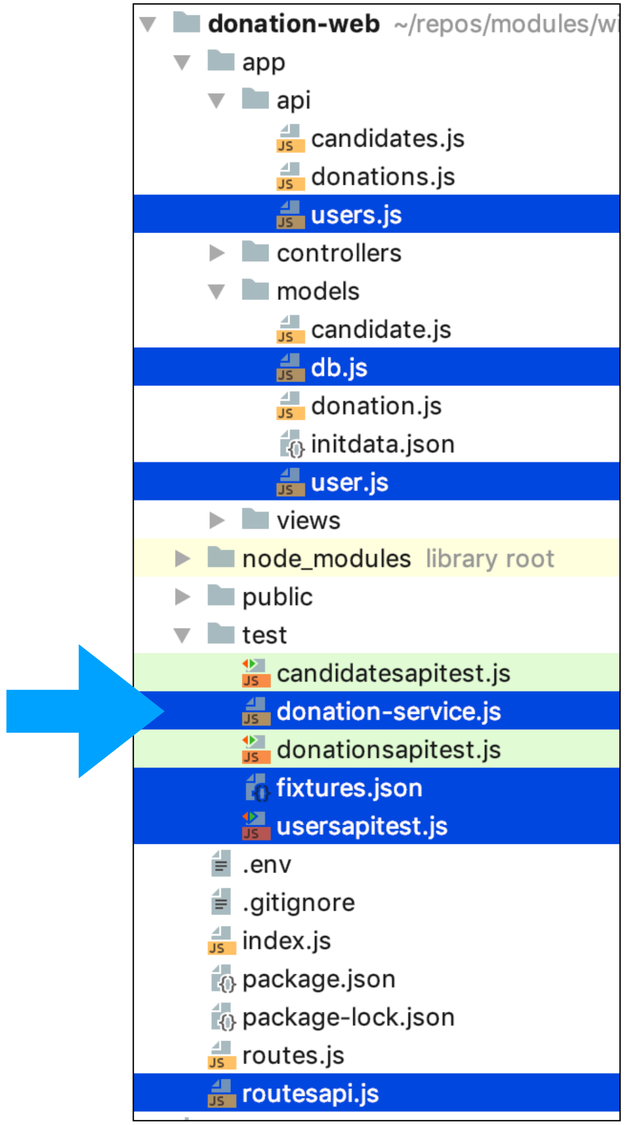
```
  deleteAll: {
    auth: false,
    handler: async function(request, h) {
      await User.deleteMany({});
      return { success: true };
    }
  },

  deleteOne: {
    auth: false,
    handler: async function(request, h) {
      const user = await User.deleteOne({ _id: request.params.id });
      if (user) {
        return { success: true };
      }
      return Boom.notFound('id not found');
    }
  }
};

module.exports = Users;
```



DonationService

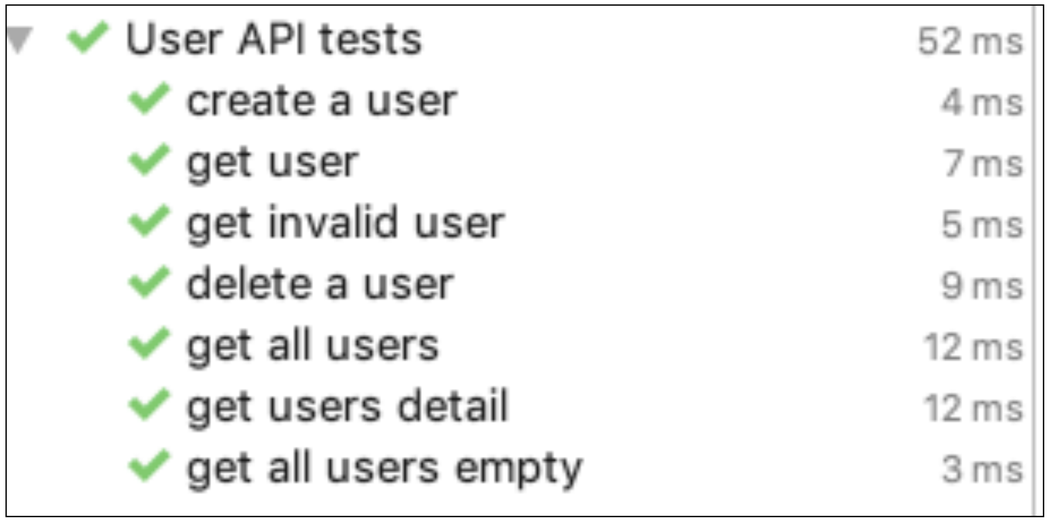
- Class to encapsulate client side access to the donation service
- Simplifies unit tests

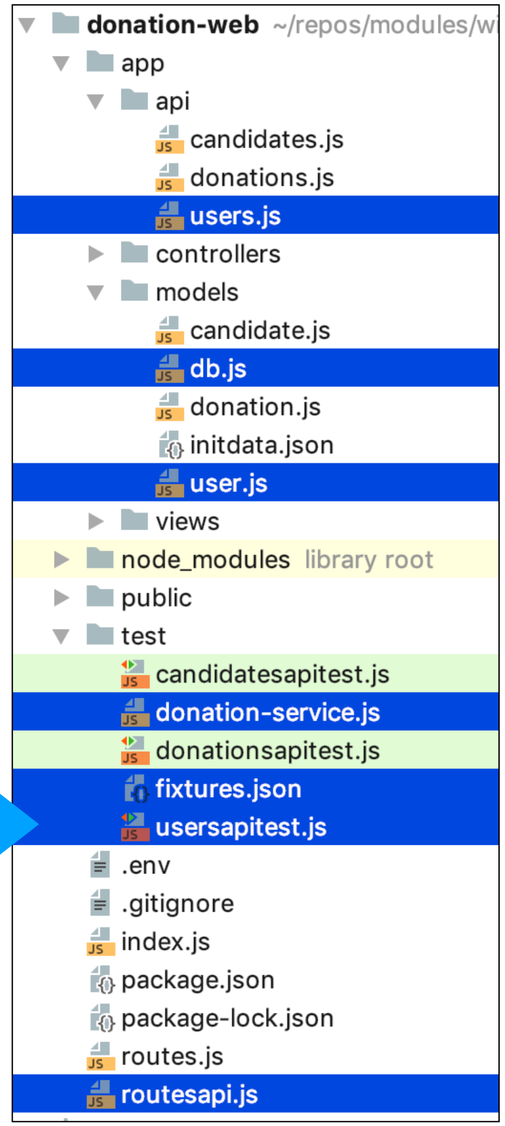
```
const axios = require('axios');

class DonationService {
  constructor(baseUrl) {
    this.baseUrl = baseUrl;
  }

  async getUsers() {
    try {
      const response = await axios.get(this.baseUrl + '/api/users');
      return response.data;
    } catch (e) {
      return null;
    }
  }

  async getUser(id) {
    try {
      const response = await axios.get(this.baseUrl + '/api/users/' + id);
      return response.data;
    } catch (e) {
      return null;
    }
  }

  async createUser(newUser) {
    try {
      const response = await axios.post(this.baseUrl + '/api/users', newUser);
      return response.data;
    } catch (e) {
      return null;
    }
  }

  async deleteAllUsers() {
    try {
      const response = await axios.delete(this.baseUrl + '/api/users');
      return response.data;
    } catch (e) {
      return null;
    }
  }

  async deleteOneUser(id) {
    try {
      const response = await axios.delete(this.baseUrl + '/api/users/' + id);
      return response.data;
    } catch (e) {
      return null;
    }
  }
```



Unit Tests

- Unit tests to verify the users endpoint
- Exercise all aspects of the endpoint.





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

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

  let users = fixtures.users;
  let newUser = fixtures.newUser;

  const donationService = new DonationService(fixtures.donationService);

  setup(async function () {
    await donationService.deleteAllUsers();
  });

  teardown(async function () {
    await donationService.deleteAllUsers();
  });

  test('create a user', async function () {
    const returnedUser = await donationService.createUser(newUser);
    assert(_.some([returnedUser], newUser), 'returnedUser must be a superset of newUser');
    assert.isDefined(returnedUser._id);
  });

  test('get user', async function () {
    const u1 = await donationService.createUser(newUser);
    const u2 = await donationService.getUser(u1._id);
    assert.deepEqual(u1, u2);
  });

  test('get invalid user', async function () {
    const u1 = await donationService.getUser('1234');
    assert.isNull(u1);
    const u2 = await donationService.getUser('012345678901234567890123');
    assert.isNull(u2);
  });
```

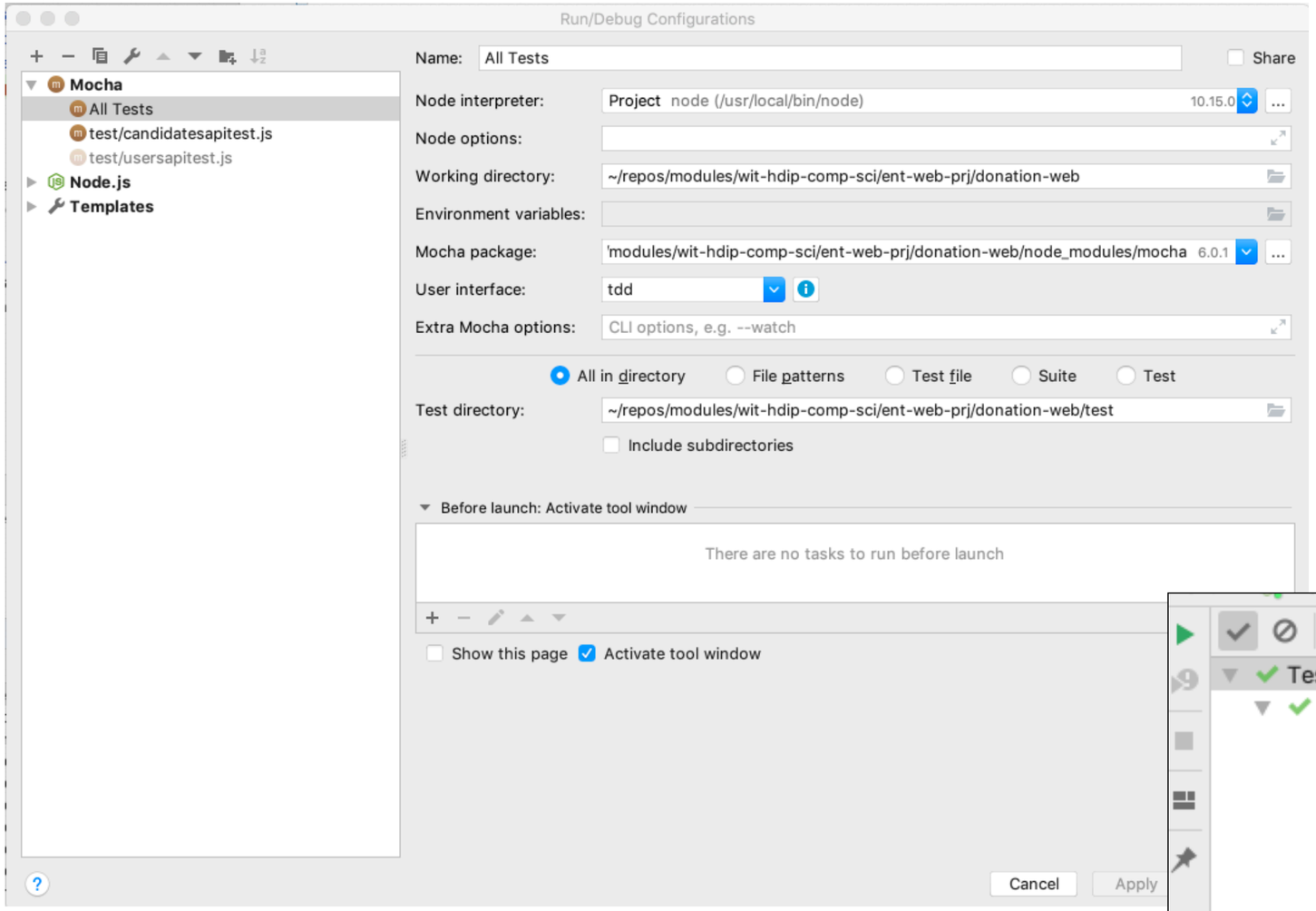
Unit Tests

- Create donationService interface (with service url loaded from fixture)
- Remove all users to ensure tests are isolated
- Tests:
  - Create
  - Get
  - Get Invalid User

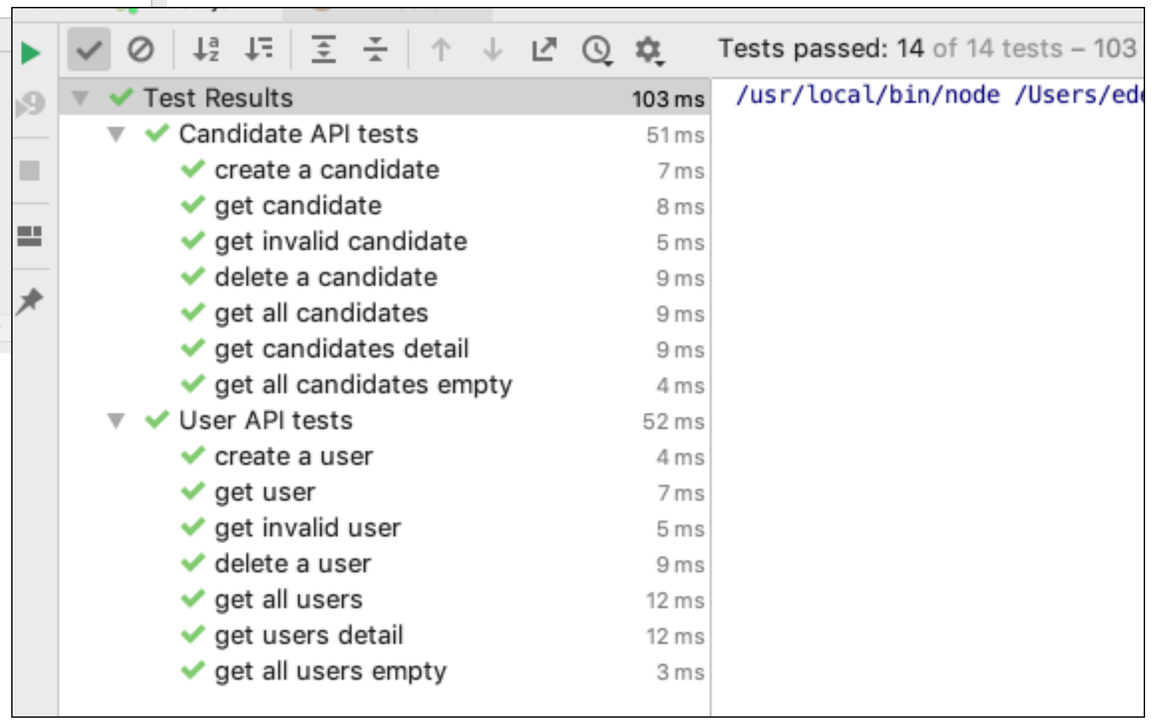
Unit Tests

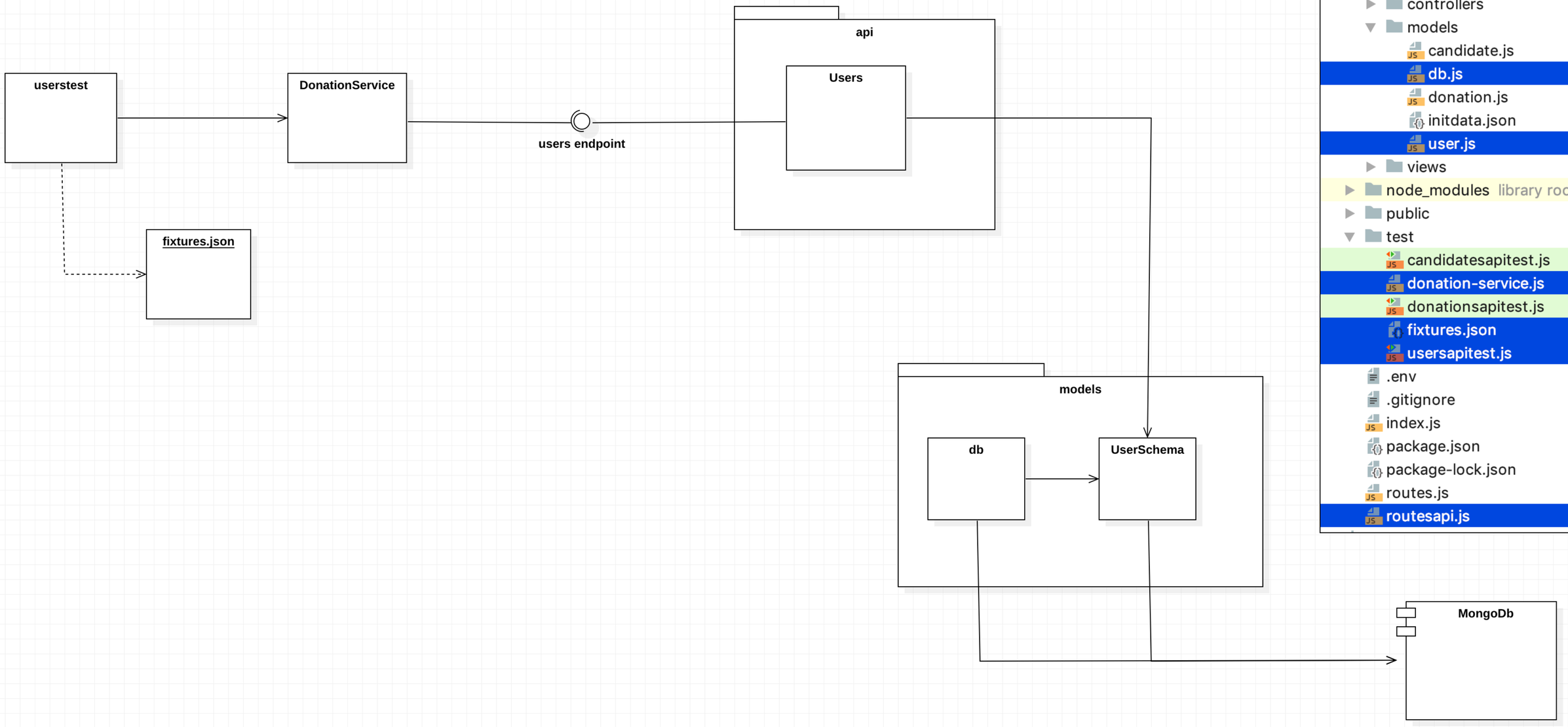
- Delete
- Get All Users
- Get Users Detail
- Get Empty user set

```
test('delete a user', async function () {
  let u = await donationService.createUser(newUser);
  assert(u._id != null);
  await donationService.deleteOneUser(u._id);
  u = await donationService.getUser(u._id);
  assert(u == null);
});

test('get all users', async function () {
  for (let u of users) {
    await donationService.createUser(u);
  }

  const allUsers = await donationService.getUsers();
  assert.equal(allUsers.length, users.length);
});

test('get users detail', async function () {
  for (let u of users) {
    await donationService.createUser(u);
  }

  const allUsers = await donationService.getUsers();
  for (var i = 0; i < users.length; i++) {
    assert(_.some([allUsers[i]], users[i]), 'returnedUser must be a superset of newUser');
  }
});

test('get all users empty', async function () {
  const allUsers = await donationService.getUsers();
  assert.equal(allUsers.length, 0);
});
```



- Comprehensive unit tests
- Run full suite




userstest
DonationService
users endpoint
api
Users
fixtures.json
models
db
UserSchema
MongoDb

donation-web ~/repos/modules/w
app
api
candidates.js
donations.js
users.js
controllers
models
candidate.js
db.js
donation.js
initdata.json
user.js
views
node_modules library root
public
test
candidatesapitest.js
donation-service.js
donationsapitest.js
fixtures.json
usersapitest.js
.env
.gitignore
index.js
package.json
package-lock.json
routes.js
routesapi.js