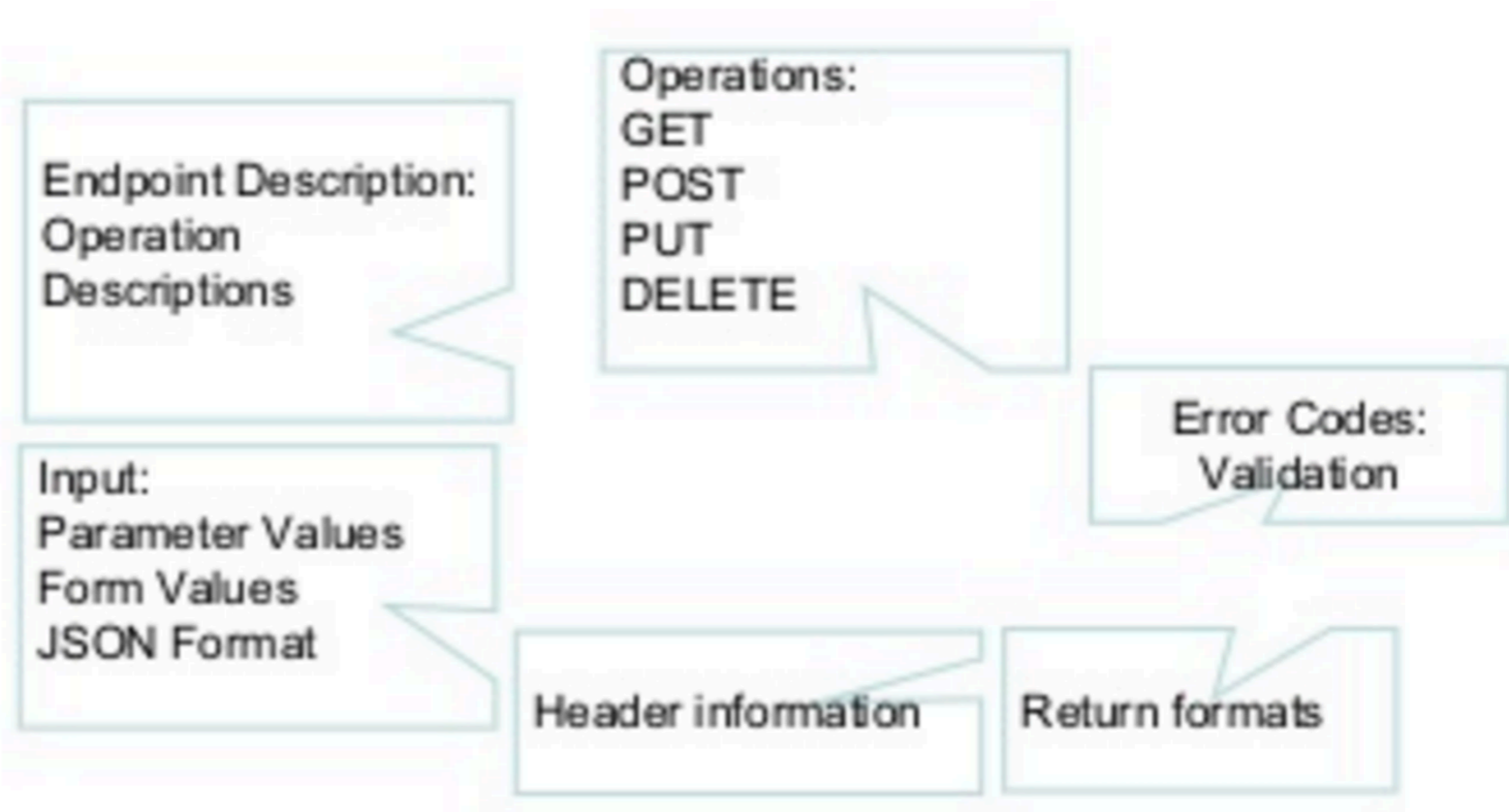


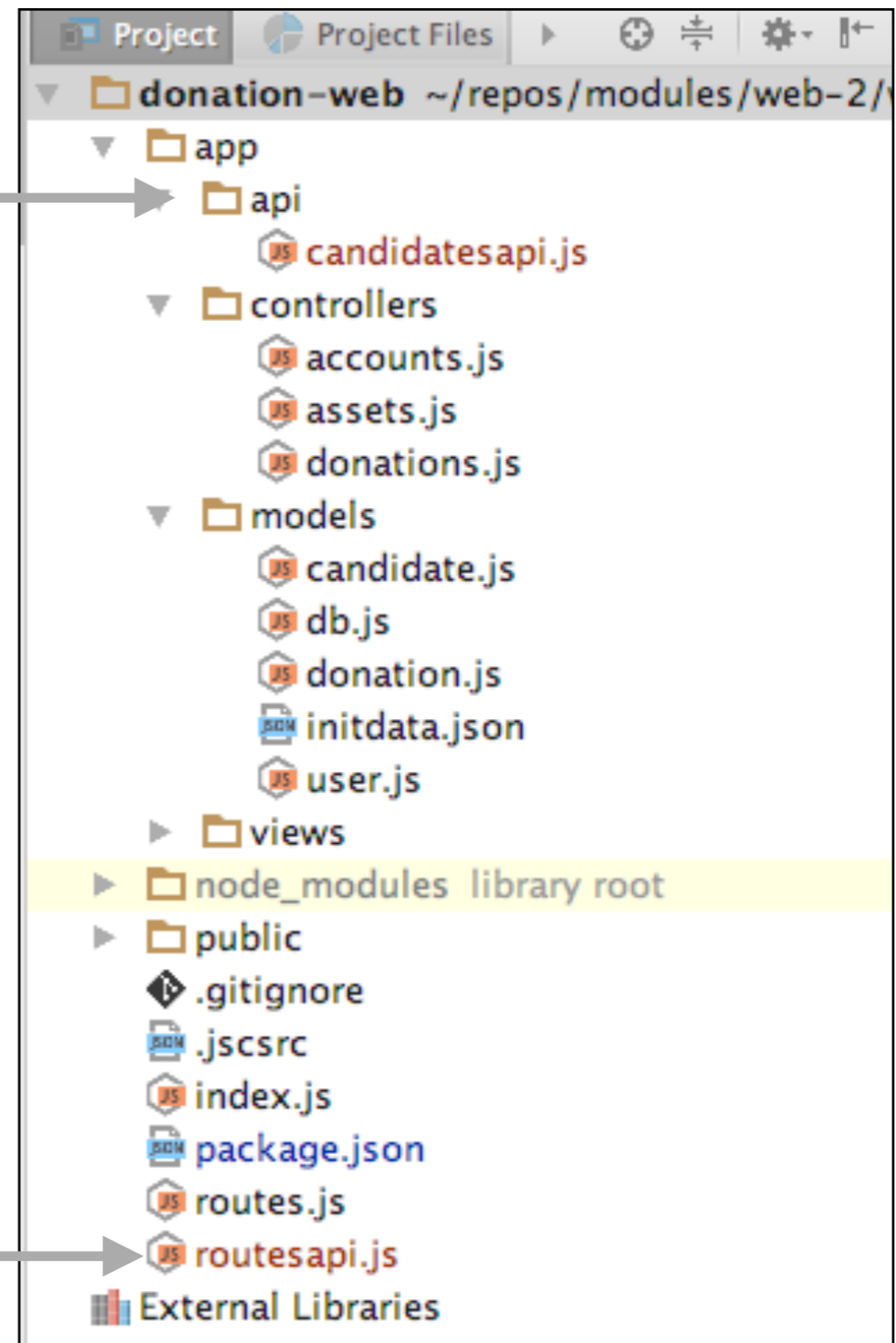
Endpoints



Project Structure

Create 'api' folder to expose endpoints from the application

Create 'routesapi.js' to define these endpoints



Get Candidates Endpoint

routesapi.js

```
const CandidatesApi = require('./app/api/candidatesapi');

module.exports = [
  { method: 'GET', path: '/api/candidates', config: CandidatesApi.find },
];
```

candidatesapi.js

```
const Candidate = require('../models/candidate');
const Boom = require('boom');

exports.find = {

  auth: false,

  handler: function (request, reply) {
    Candidate.find({}).exec().then(candidates => {
      reply(candidates);
    }).catch(err => {
      reply(Boom.badImplementation('error accessing db'));
    });
  },
};
```

localhost:4000/api/candidates x Eamonn

localhost:4000/api/candidates

+ - View source

```
[
  - {
    _id: "57b6b94d701cce8539bddea5",
    firstName: "Lisa",
    lastName: "Simpson",
    office: "President",
    __v: 0
  },
  - {
    _id: "57b6b94d701cce8539bddea6",
    firstName: "Donald",
    lastName: "Simpson",
    office: "President",
    __v: 0
  }
]
```

[0]



boom

<https://github.com/hapijs/boom>

boom provides a set of utilities for returning HTTP errors. Each utility returns a `Boom` error response object (instance of `Error`) which includes the following properties:

- `isBoom` - if `true`, indicates this is a `Boom` object instance.
- `isServer` - convenience bool indicating status code ≥ 500 .
- `message` - the error message.
- `output` - the formatted response. Can be directly manipulated after object construction to return a custom error response.

Allowed root keys:

- `statusCode` - the HTTP status code (typically 4xx or 5xx).
- `headers` - an object containing any HTTP headers where each key is a header name and value is the header content.
- `payload` - the formatted object used as the response payload (stringified). Can be directly manipulated but any changes will be lost if `reformat()` is called. Any content allowed and by default includes the following content:
 - `statusCode` - the HTTP status code, derived from `error.output.statusCode`.
 - `error` - the HTTP status message (e.g. 'Bad Request', 'Internal Server Error') derived from `statusCode`.
 - `message` - the error message derived from `error.message`.
- inherited `Error` properties.

Get Candidate (singular) Endpoint

routesapi.js

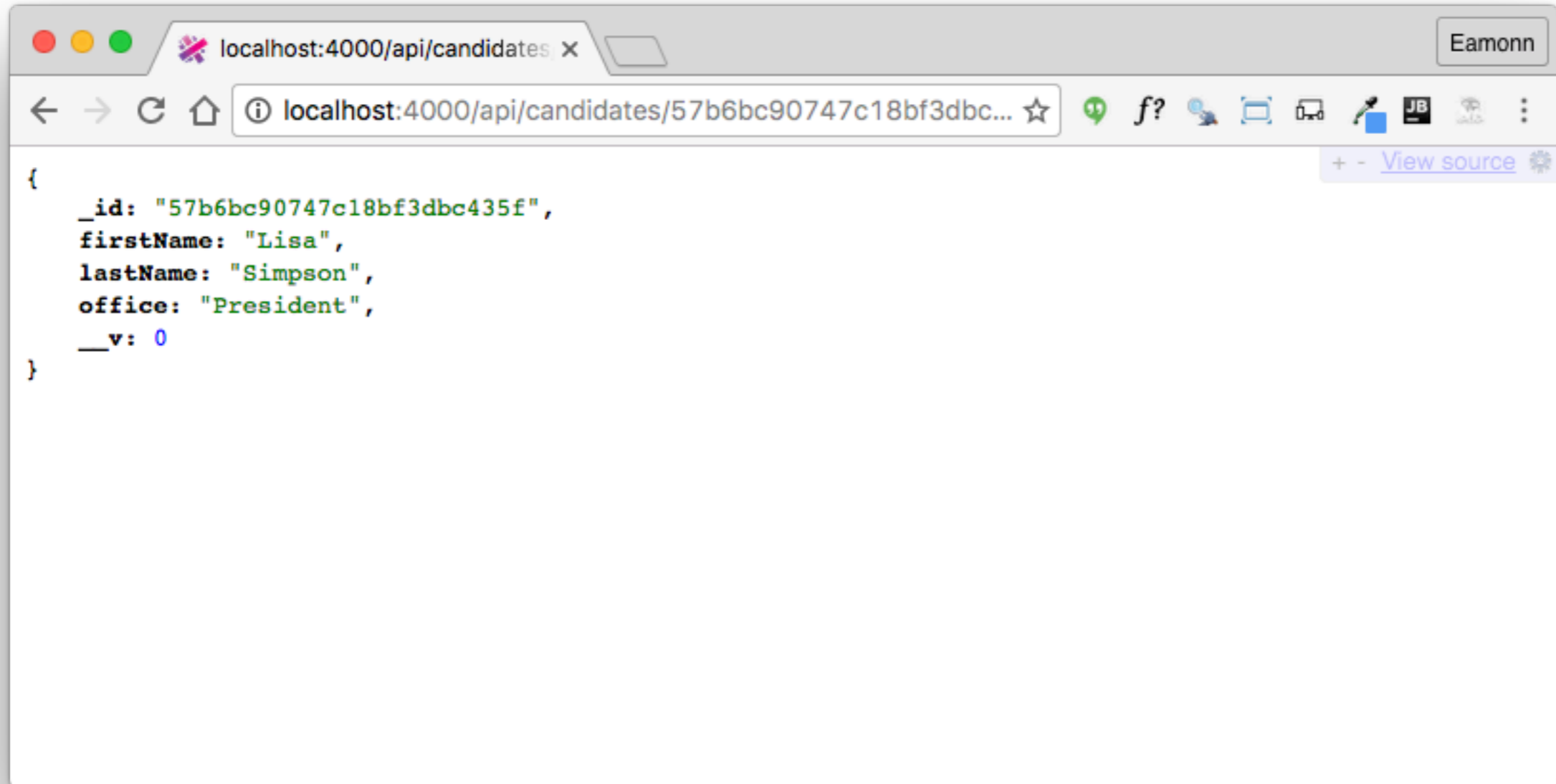
```
{ method: 'GET', path: '/api/candidates/{id}', config: CandidatesApi.findOne },
```

- Object ID passed as part of path

candidatesapi.js

```
const Candidate = require('../models/candidate');
exports.findOne = {
  auth: false,
  handler: function (request, reply) {
    Candidate.findOne({ _id: request.params.id }).then(candidate => {
      reply(candidate);
    }).catch(err => {
      reply(Boom.notFound('id not found'));
    });
  },
}
```

Get Candidate (singular) Endpoint



The screenshot shows a web browser window with a REST client interface. The address bar displays the URL `localhost:4000/api/candidates/57b6bc90747c18bf3dbc...`. The response body is a JSON object representing a candidate:

```
{
  _id: "57b6bc90747c18bf3dbc435f",
  firstName: "Lisa",
  lastName: "Simpson",
  office: "President",
  __v: 0
}
```

The browser's developer tools are open, showing the response in a code editor. The name of the browser is "Eamonn".

Boom Error Messages



The image shows a web browser window with the following details:

- Tab: localhost:4000/api/candidates
- Address Bar: localhost:4000/api/candidates/57b6bc90747c18bf3d2...
- Page Content: A JSON object representing an error response:

```
{  statusCode: 404,  error: "Not Found",  message: "id not found"}
```
- Footer: A purple bar with the text "error".

Exercising Endpoints: Postman

The screenshot displays the Postman application interface. The top navigation bar includes 'Runner', 'Import', 'Builder', and 'Team Library'. The main workspace is divided into three sections:

- Left Panel (History/Collections):** Shows a search bar and a list of requests under 'Today'. Two requests are visible: a GET request to `localhost:4000/api/candidates` and another GET request to `localhost:4000/api/candidates/57b6bc90747c18bf3dbc435f`.
- Center Panel (Request Builder):** Shows a GET request to `localhost:4000/api/candidates` with the 'Authorization' tab selected. The 'Type' is set to 'No Auth'. Buttons for 'Send' and 'Save' are visible.
- Right Panel (Response):** Shows the response body in 'Pretty' view. The status is '200 OK', the time is '28 ms', and the size is '438 B'. The response is a JSON array of two objects:

```
1 [
2   {
3     "_id": "57b6bc90747c18bf3dbc435f",
4     "firstName": "Lisa",
5     "lastName": "Simpson",
6     "office": "President",
7     "__v": 0
8   },
9   {
10    "_id": "57b6bc90747c18bf3dbc4360",
11    "firstName": "Donald",
12    "lastName": "Simpson",
13    "office": "President",
14    "__v": 0
15  }
16 ]
```

Postman

Runner Import Builder Team Library SYNC OFF Eamonn d... No environment

Search

History Collections

All Me Team

donation-web 2 requests

GET get all candidates

GET get one candidate

Postman Echo 21 requests

get all candidates get one candidate

GET localhost:4000/api/candidates/57b6bc90747c18bf3 Params Send Save

Authorization Headers Body Pre-request Script Tests Manage Cookies Generate Code

Type No Auth

Body Cookies Headers (7) Tests Status: 200 OK Time: 40 ms Size: 330 B

Pretty Raw Preview JSON Save Response

```
1 {
2   "_id": "57b6bc90747c18bf3dbc435f",
3   "firstName": "Lisa",
4   "lastName": "Simpson",
5   "office": "President",
6   "__v": 0
7 }
```