



Global Knowledge®

ilionx

# Module – Towards a microservice architecture

## WORLDWIDE LOCATIONS

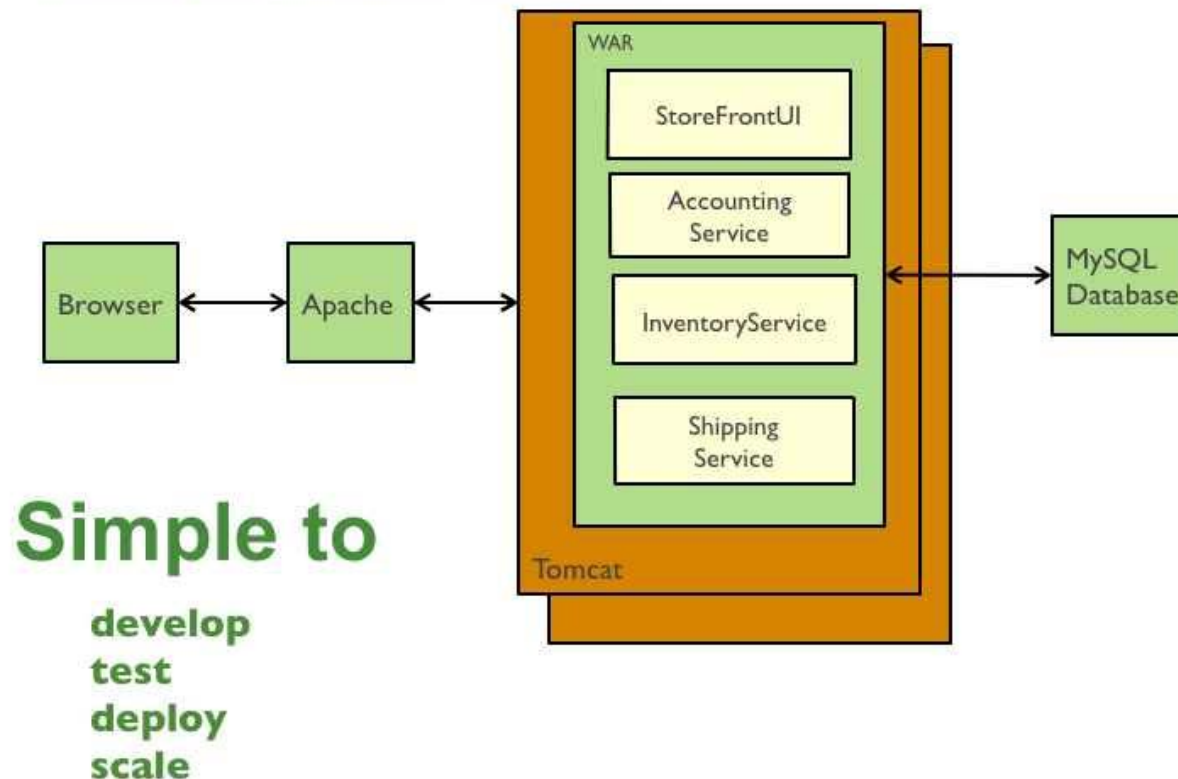
BELGIUM CANADA COLOMBIA DENMARK EGYPT FRANCE IRELAND JAPAN KOREA MALAYSIA MEXICO NETHERLANDS NORWAY QATAR  
SAUDI ARABIA SINGAPORE SPAIN SWEDEN UNITED ARAB EMIRATES UNITED KINGDOM UNITED STATES OF AMERICA

# What is a microservice architecture?

Traditionally – **monolithic** architecture

This is **not BAD** per se, just a different (slightly older) approach

Traditional web application architecture



*“For example, consider a monolithic ecommerce SaaS application. It might contain a web server, a load balancer, a catalog service that services up product images, an ordering system, a payment function, and a shipping component.”*

# Possible problems w/ monolithic approach

Recompilation and deployment of the complete application, even with small changes

Structure & architecture gets complicated

Errors in one part of the app can influence the rest of the app

...

*Microservices to the rescue*

*“Service Oriented  
Architecture (SOA)”*

### Monolithic Architecture



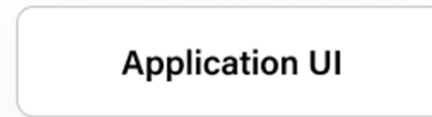
### Business Logic



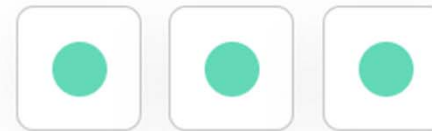
### Databases



### Microservices



### Business Logic



### Databases



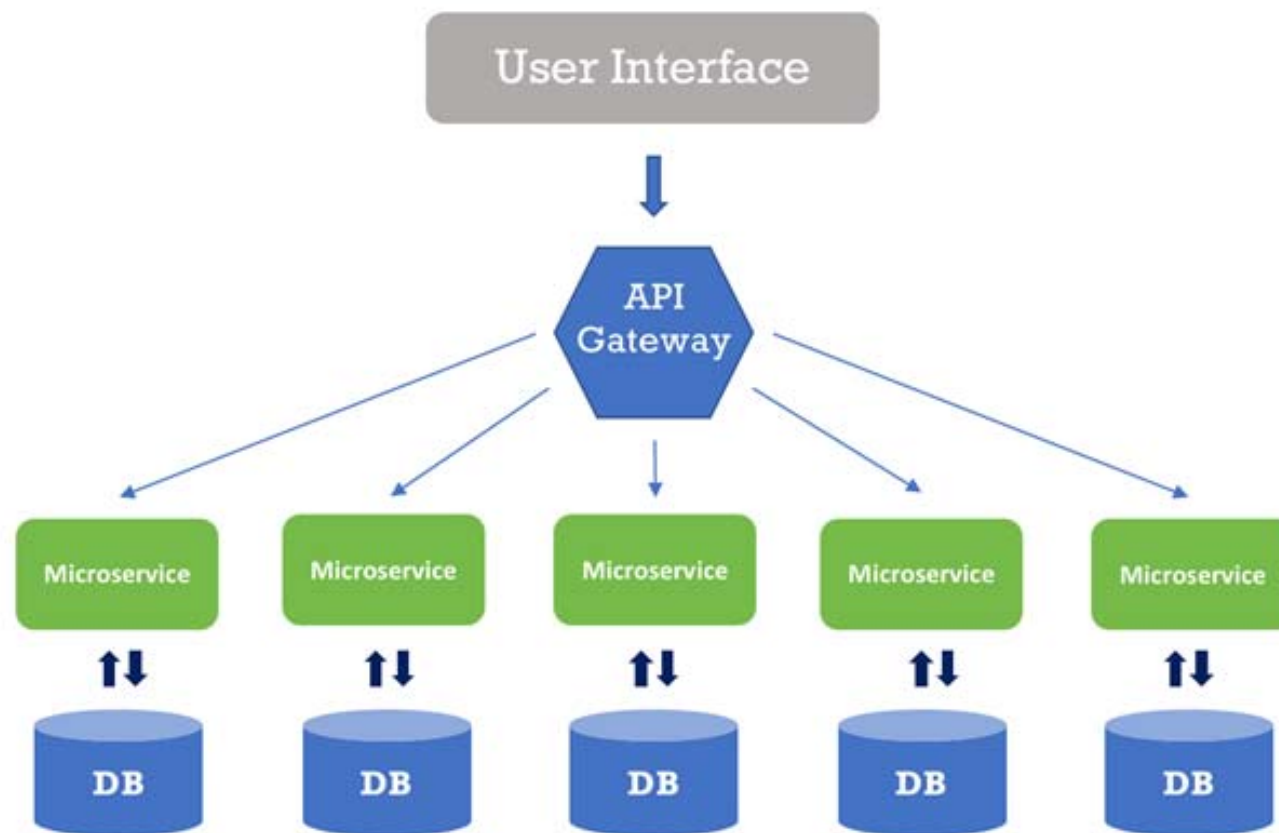
# Microservice approach

Microservices are a style of service-oriented architecture (SOA) where the app is structured on an **assembly of interconnected services**.

With microservices, the application architecture is built with **lightweight** protocols.

The services are **finely seeded** in the architecture.

Microservices **disintegrate the app** into smaller services and enable improved modularity





# Advantages

greater flexibility,

high scalability,

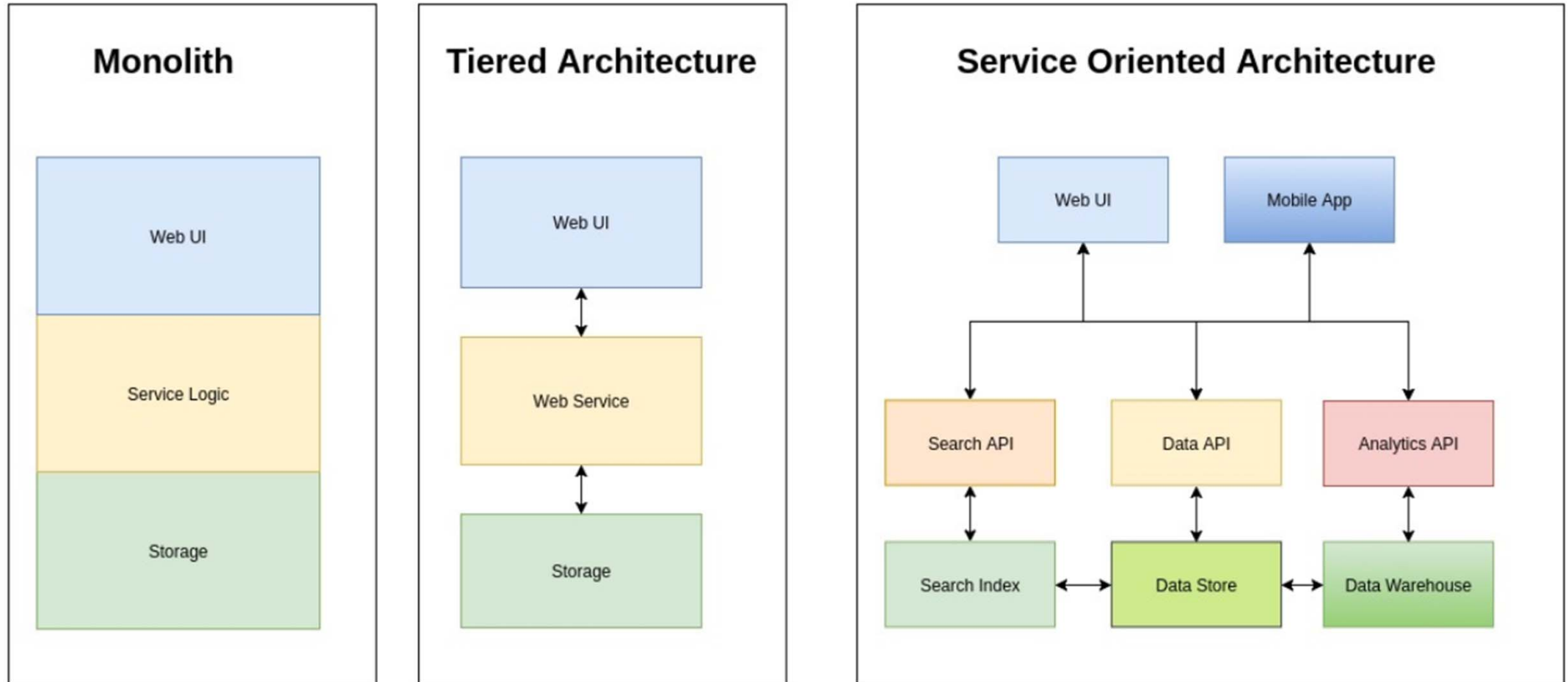
continuous development & -deployment,

systematic data organization,

time optimization,

reliability.

# Evolution of application architecture



# Downsides of SOA

Are there any? Hell, yeah.

Possible overly complicated and fragmentated system

Monitoring complete application can be difficult – multiple points of failure

New developers are easily overwhelmed / discovery of structure is (more) complicated

# Allright – so how to build SOA?

Choice: using Node.js + Express here.

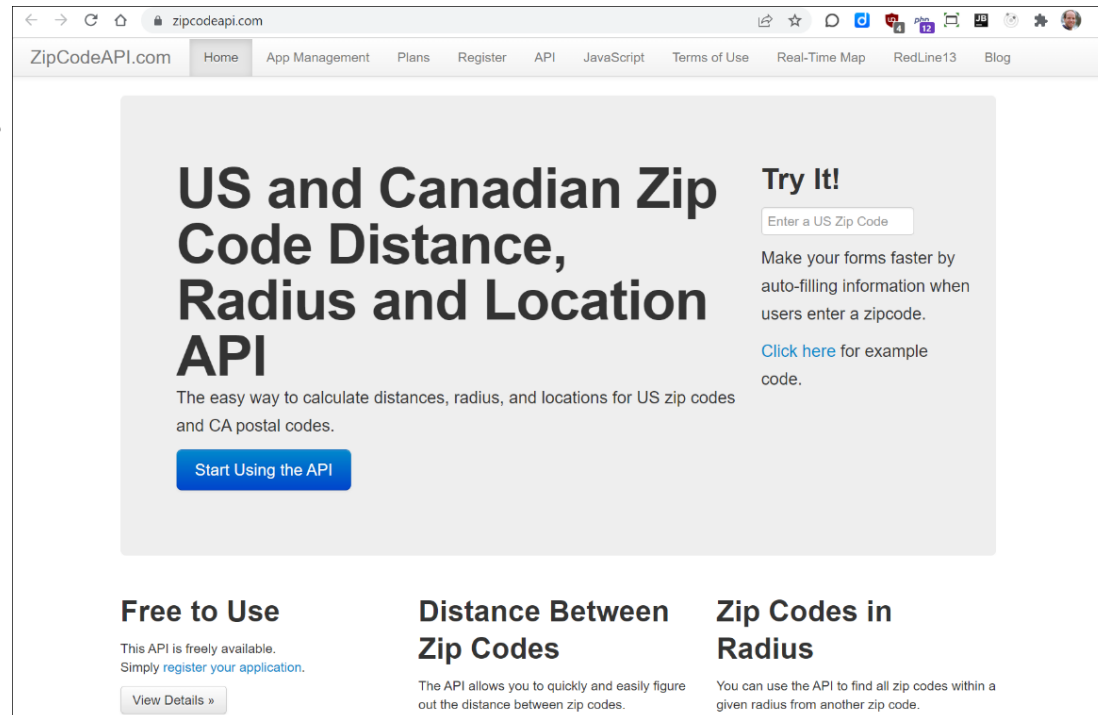


express

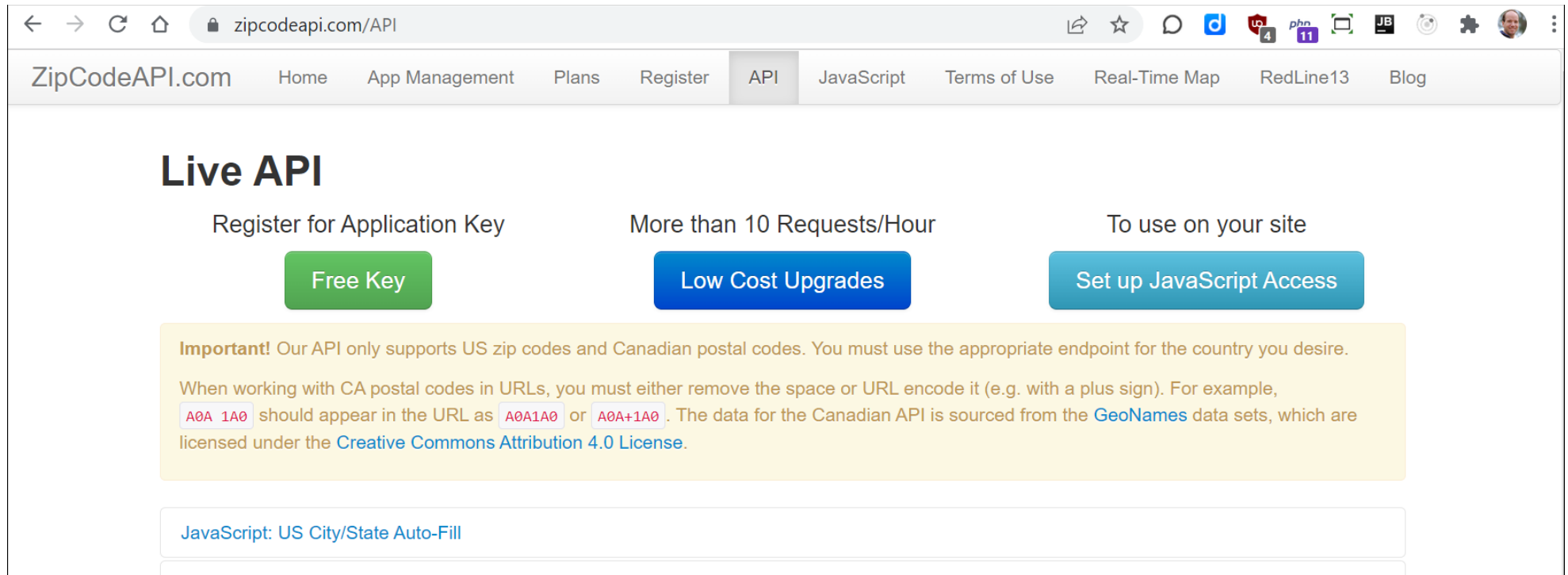
<https://expressjs.com/>

# Step 1/5 – developing or choosing backend/API

- This example: using [ZipCodeApi.com](https://zipcodeapi.com/)
- Free, but register w/ email for a key
- ‘Measure the distance between US Zipcodes’
- We could have used Google Places API for that
  - but, more code/more difficult –
  - this example uses an API that does exactly one (1) thing.
- There are a [TON](#) of other (free) API's available!  
Spotify, LastFM, Twitter, bol.com, etc



# Register your 'application'



The screenshot shows the ZipCodeAPI.com website with the 'API' tab selected in the navigation bar. The main heading is 'Live API'. Below it are three columns of options: 'Register for Application Key' with a 'Free Key' button, 'More than 10 Requests/Hour' with a 'Low Cost Upgrades' button, and 'To use on your site' with a 'Set up JavaScript Access' button. A yellow warning box contains important information about US zip codes and Canadian postal codes, including URL encoding examples. At the bottom, there is a link for 'JavaScript: US City/State Auto-Fill'.

ZipCodeAPI.com Home App Management Plans Register API JavaScript Terms of Use Real-Time Map RedLine13 Blog

## Live API

Register for Application Key      More than 10 Requests/Hour      To use on your site

[Free Key](#)      [Low Cost Upgrades](#)      [Set up JavaScript Access](#)

**Important!** Our API only supports US zip codes and Canadian postal codes. You must use the appropriate endpoint for the country you desire.

When working with CA postal codes in URLs, you must either remove the space or URL encode it (e.g. with a plus sign). For example, `A0A 1A0` should appear in the URL as `A0A1A0` or `A0A+1A0`. The data for the Canadian API is sourced from the [GeoNames](#) data sets, which are licensed under the [Creative Commons Attribution 4.0 License](#).

[JavaScript: US City/State Auto-Fill](#)

You can also use the provided (=my) key, but you'll run into limitations when testing

## Step 2/5 – setting up your server

```
// server.js  
// 1. Import and create express() application  
const express = require('express');  
const app = express();  
  
// 2. Import routes and pass in the created app to bind the  
// routes to the app.  
const routes = require('./api-routes/routes');  
routes(app);  
  
// 3. Create a port to listen to  
const port = process.env.PORT || 3000;  
  
// 4. Start the server on the given port  
app.listen(port, () => {  
    console.log(`Listening on port http://localhost:${port}`);  
});
```

## Step 3/5 – specify valid routes

```
// routes.js
'use strict';

// 1. Import the controller, to specify the actions on
// requested routes
const controller = require('../controllers/controller');

// 2. Enhance the passed in application with the routes
module.exports = app => {
  app.route('/about')
    .get(controller.about);

  // 2a. dynamic parameters, like http://localhost:3000/distance/10001/90001
  //      (= New York to Los Angeles)
  app.route('/distance/:zipcode1/:zipcode2')
    .get(controller.getDistance);
}
```



## Step 4/5 – build the controller

```
// controller.js
'use strict';

// 1. Import the correct files
const properties = require('../package.json');
const distance = require('../service/distance');

// 2. Specify methods on this controller
var controllers = {
  // 2a. Use package.json to show information.
  about: (req, res) => {
    var aboutInfo = {
      name: properties.name,
      version: properties.version,
    }
    res.json(aboutInfo);
  },

  // 2b. Use the imported distance object
  // to calculate actual distance and send to client.
  getDistance: (req, res) => {
    distance.find(req, res, (err, dist) => {
      if (err)
        res.send(err);
      res.json(dist);
    });
  },
};

// 3. export the controller
module.exports = controllers;
```

# Step 5/5 – build the communication w/ API

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

// 2. My key - Please register for your own key. You can store this in an environment variable for safety.
const apiKey = process.env.ZIPCODE_API_KEY || "sSHNTiMLwwIUItZWxexJMVC7G2...";

// 3. The API endpoint to talk to. You can store this in an environment variable for safety.
const zipCodeURL = process.env.ZIPCODE_API_URL || 'https://www.zipcodeapi.com/rest/';

// 4. The exported object with external API communication function (called 'find')
var distance = {
  find: (req, res, next) => {
    request(zipCodeURL + apiKey +
      '/distance.json/' + req.params.zipcode1 + '/' +
      req.params.zipcode2 + '/km', // 4b. you can also calculate 'mile' if you want to
    (error, response, body) => {
      if (!error && response.statusCode === 200) {
        response = JSON.parse(body);
        res.send(response);
      } else {
        console.log(response.statusCode + response.body);
        res.send({
          distance: -1
        });
      }
    })
  }
};

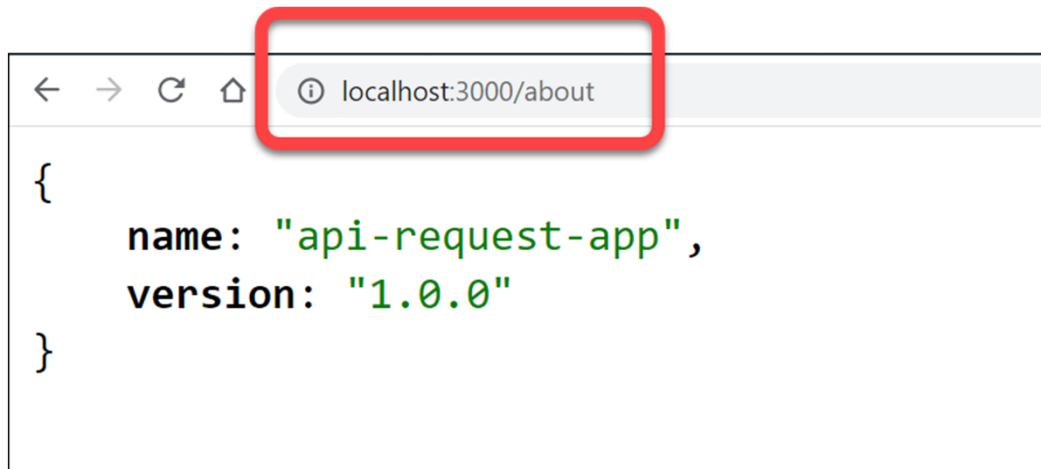
// 5. export the object w/ function
module.exports = distance;
```

# Test your service



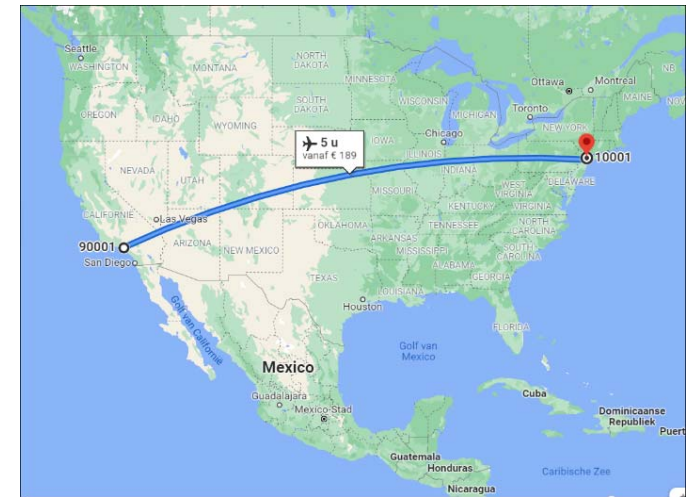
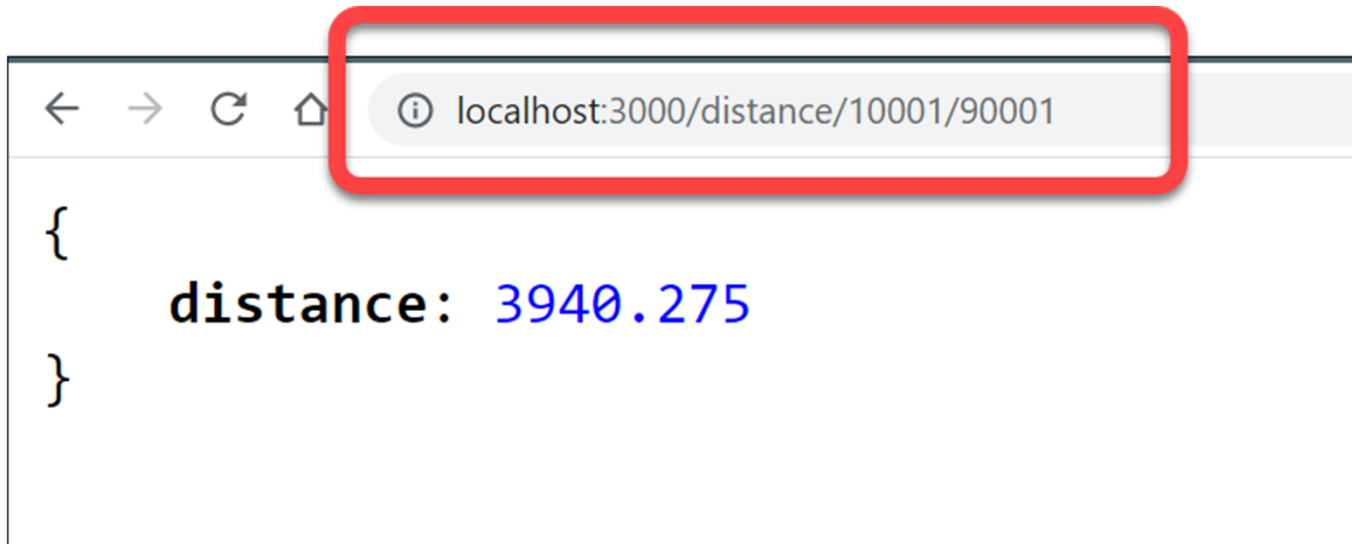
A terminal window with a dark background. The command `> api-request-app 1.0.0 start C:\Users\Gebruiker\Desktop\request-app` is entered. Below it, the command `> node server.js` is entered and highlighted with a red rectangle. The output shows the server listening on port <http://localhost:3000>.

```
> api-request-app 1.0.0 start C:\Users\Gebruiker\Desktop\request-app  
> node server.js  
Listening on port http://localhost:3000
```



A web browser window with a light gray header. The address bar shows `localhost:3000/about` and is highlighted with a red rectangle. The main content area displays a JSON object: `{ name: "api-request-app", version: "1.0.0" }`.

```
{  
  name: "api-request-app",  
  version: "1.0.0"  
}
```



Very simple. Calculate distance (in km) between New York (10001) and Los Angeles (90001)

# Build your Frontend

We're using **Angular + Bootstrap** here, but any frontend will do!

- Start server + start frontend

NgRequest

localhost:4200

## Measure distance between US Zipcodes

Enter two valid US Zipcodes:

10001

90001

Get distance Clear

### Some example cities/zipcodes

City	Zip
New York	10001-10090
Chicago	60007-60027
Los Angeles	90001-90090
Seattle	98101-98117
Denver	80014-80211
Miami	33101-33136

### Distance

Distance between **10001** and **90001** in kilometer:

**3940.275**

# Example code

The screenshot displays the GitHub interface for the repository **PeterKassenaar / ilionx**. The repository is public and has 0 stars, 0 forks, and 1 watcher. The commit history shows three recent commits, all titled "Added example code", with the file **code** highlighted. The README file is open, showing the title **ilionx** and the description "Slides and demo code on the Inspiration Session, Ilionx - spring 2022". The right sidebar shows repository statistics and a language usage chart.

**Repository Statistics:**

- Unwatch: 1
- Fork: 0
- Star: 0

**Commit History:**

Commit	Message	Time
48f912a	Added example code	13 seconds ago
	Added example code	13 seconds ago
	Added example code	13 seconds ago

**Language Usage:**

Language	Percentage
TypeScript	44.2%
JavaScript	34.7%

*Questions?*