Global Knowledge®

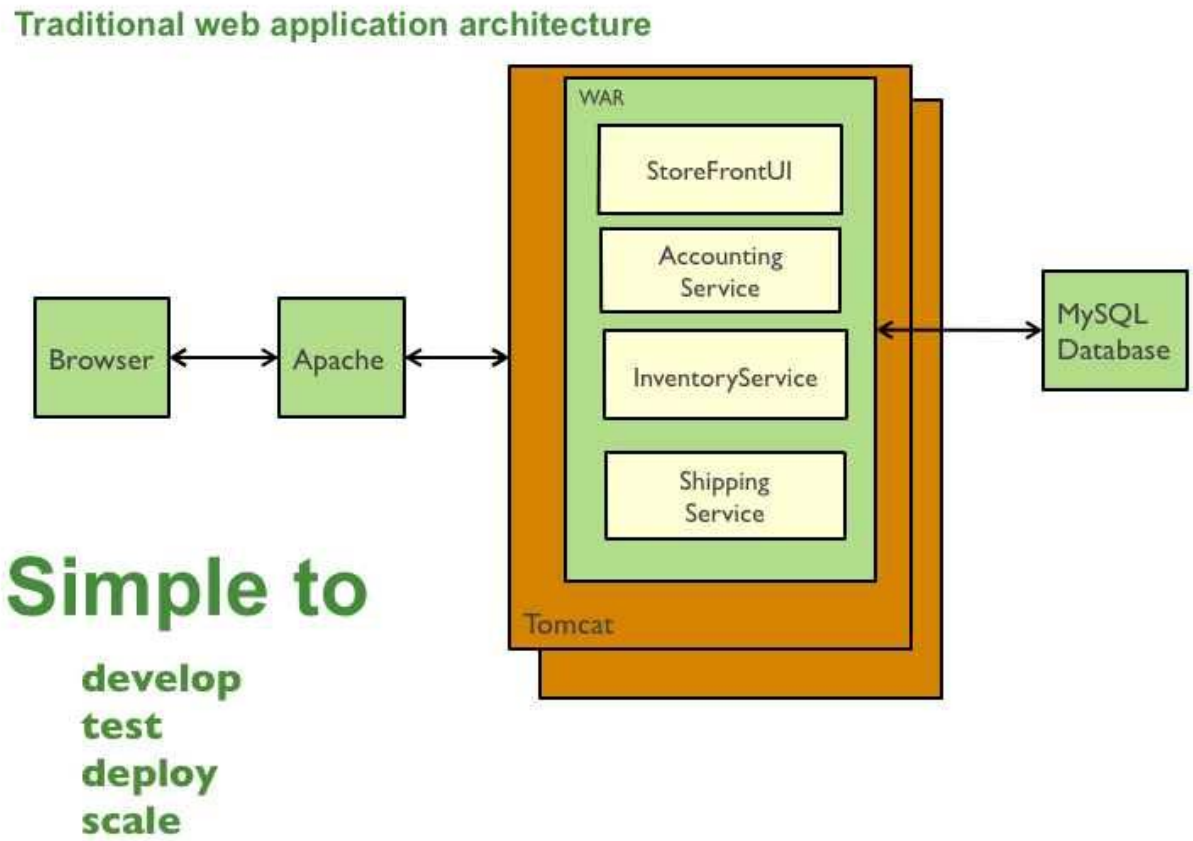
ilionx

Module – Towards a microservice architecture

What is a microservice architecture?

Traditionally – monolithic architecture

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



https://microservices.io/patterns/monolithic.html

"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

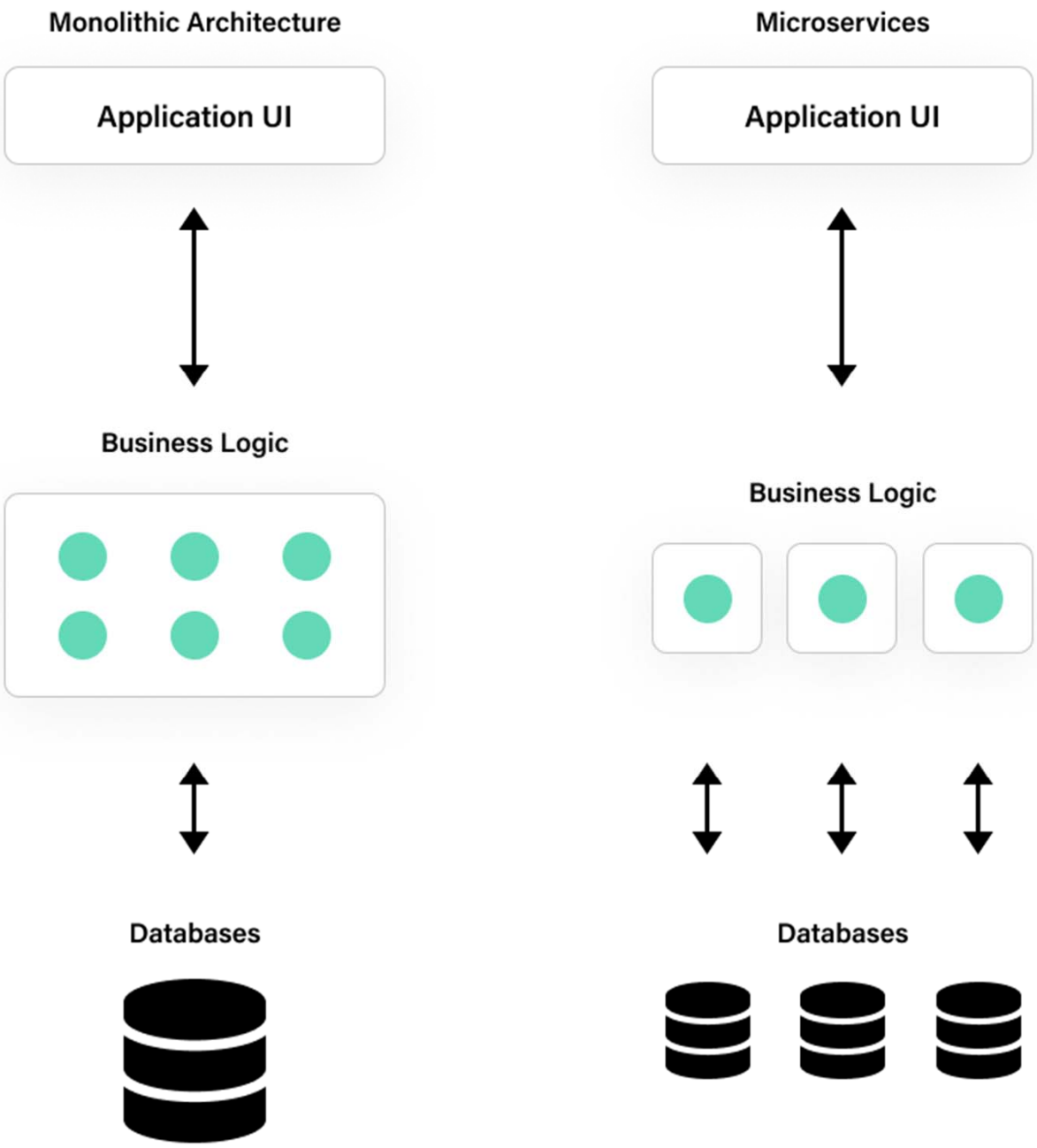
Structure & architecture gets complicated

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

...

Microservices to the rescue

“Service Oriented Architecture (SOA)”



https://fauna.com/blog/how-to-build-microservices-with-node-js

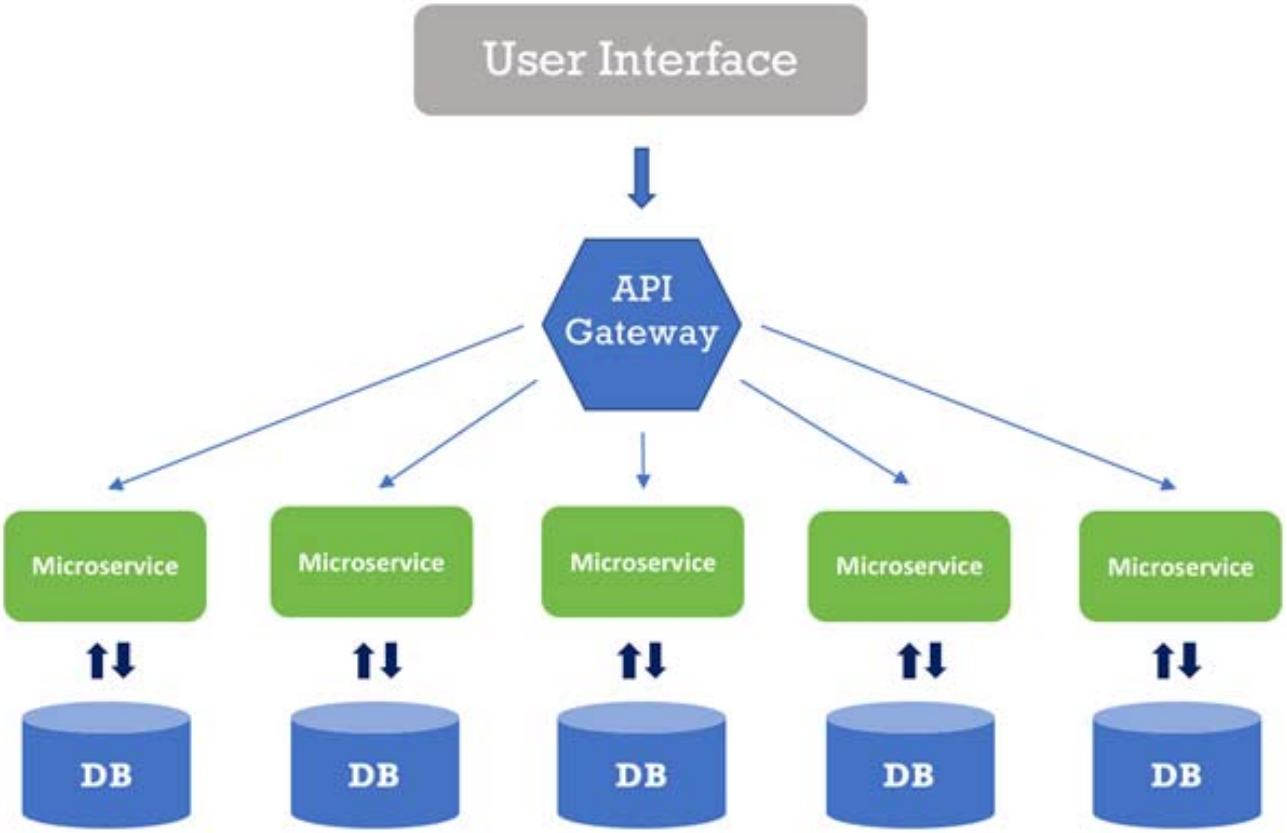
Microservice approach

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



https://frontegg.com/blog/implementing-microservices-in-nodejs

Advantages

greater flexibility,

high scalability,

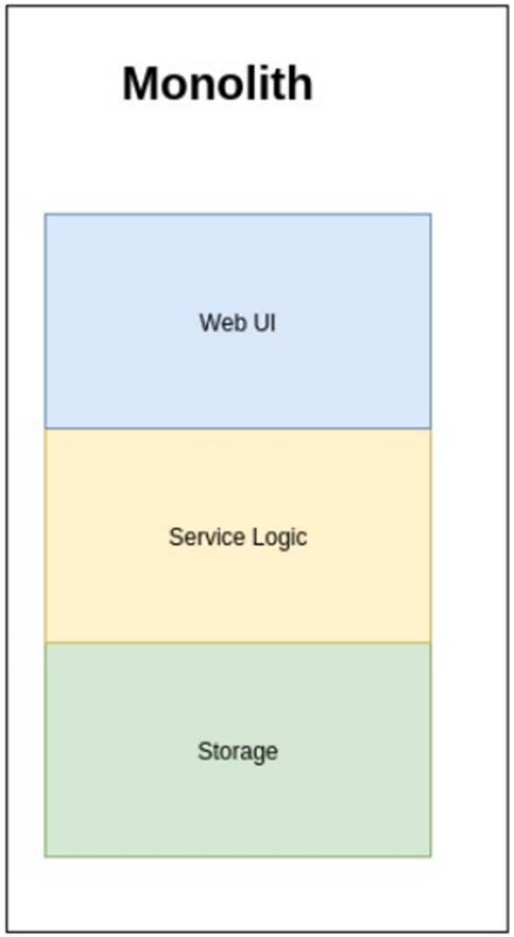
continuous development & -deployment,

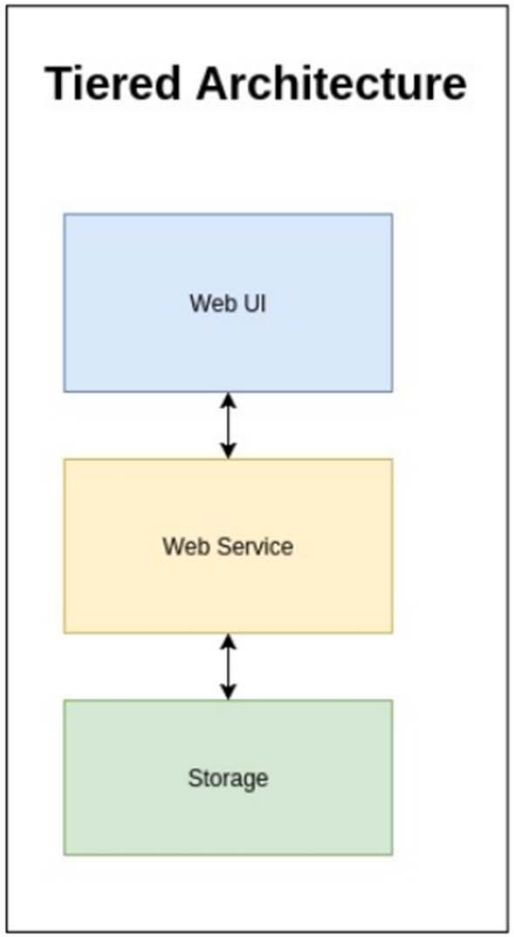
systematic data organization,

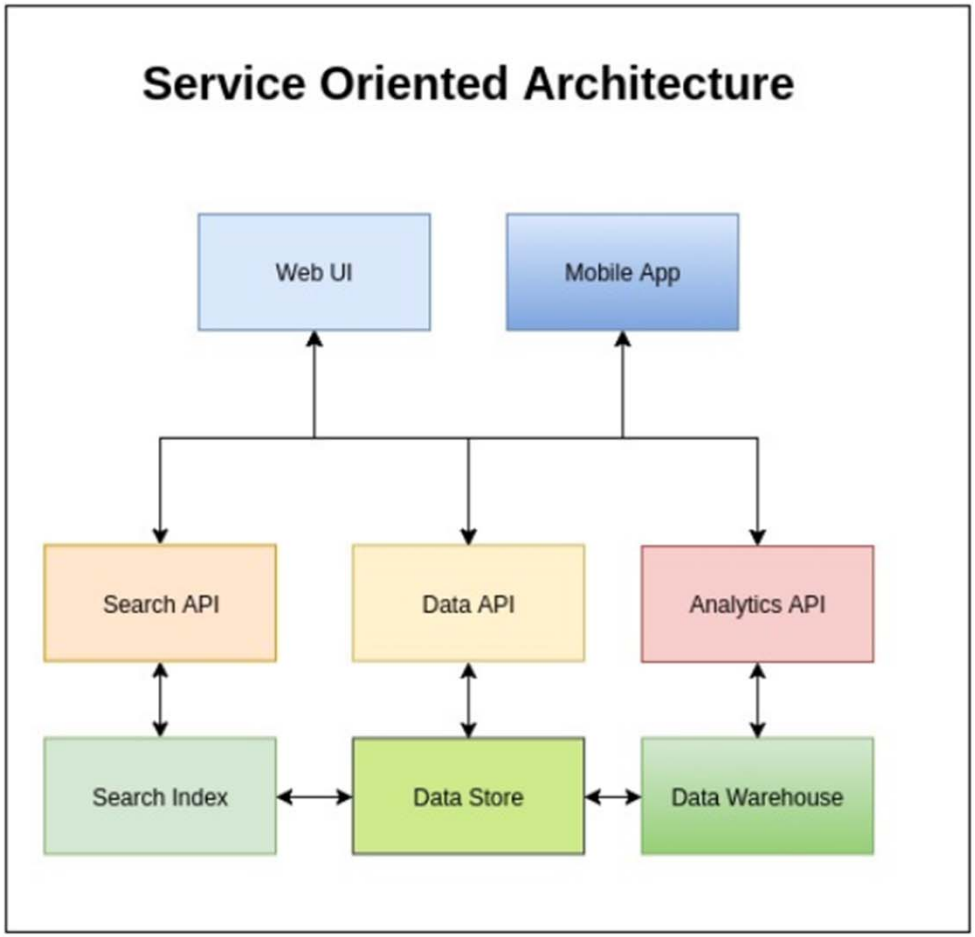
time optimization,

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.

https://nodejs.org/en/

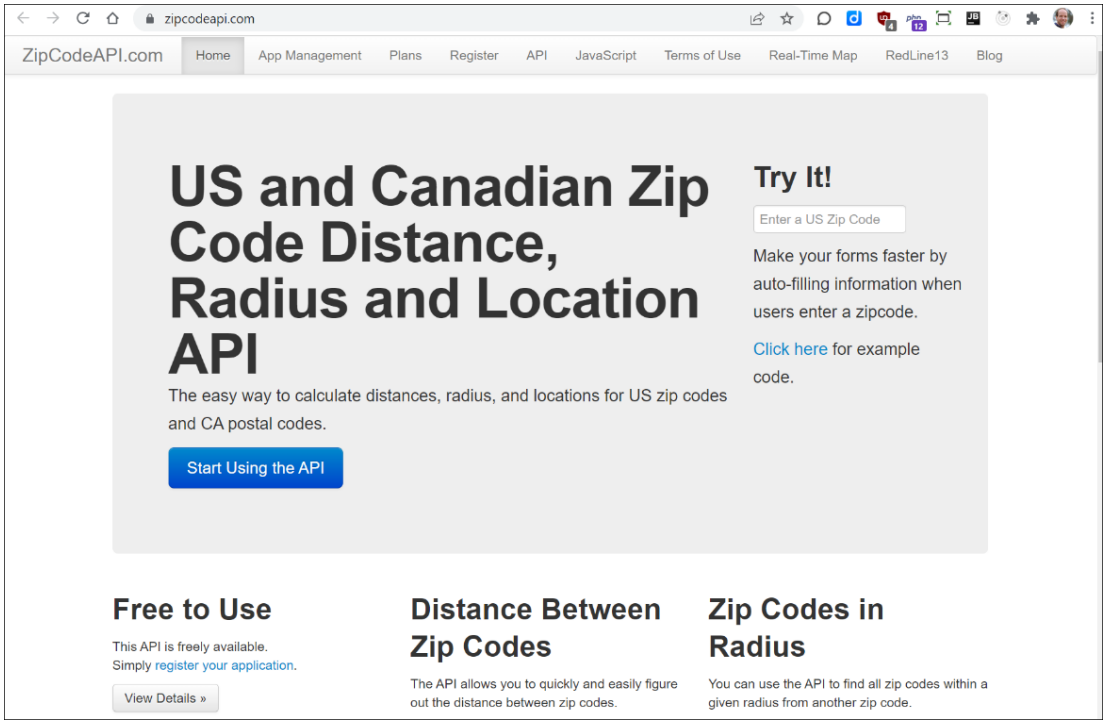
express

https://expressjs.com/

Step 1/5 – developing or choosing backend/API

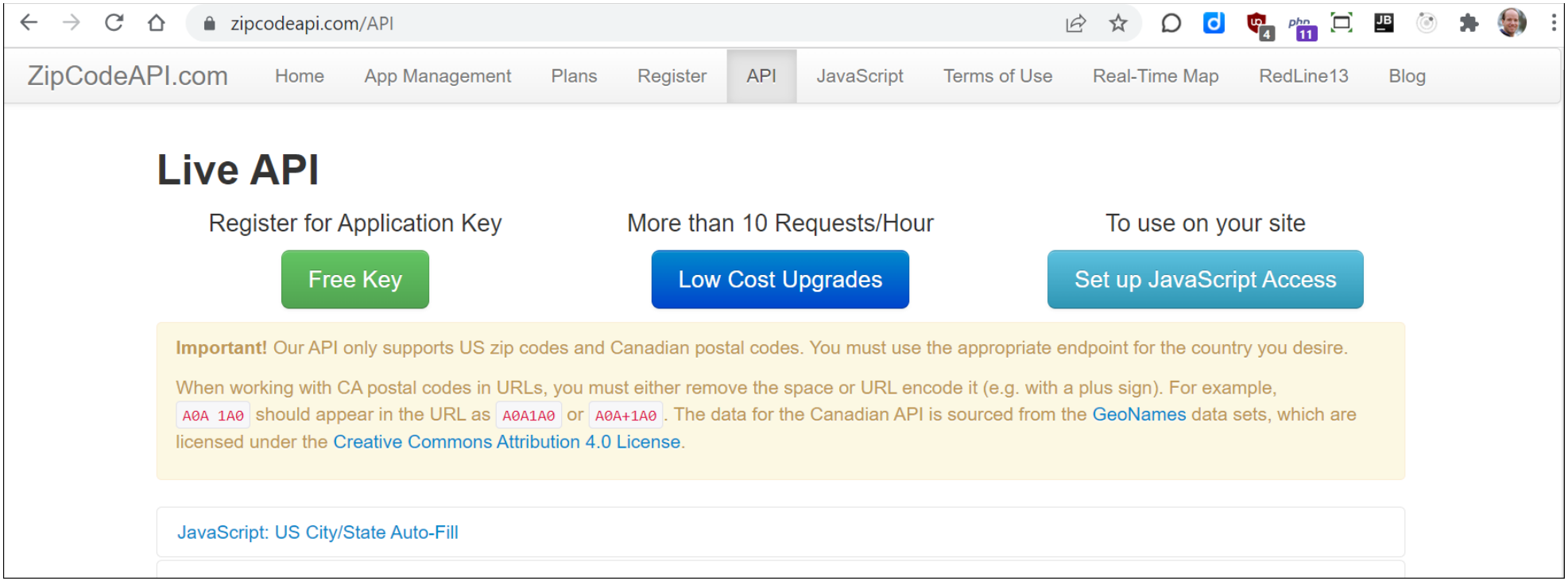
- This example: using ZipCodeApi.com
- Free, but register w/ email for a key
- 'Measure the distance between US Zipcodes'
- We could have used Google Places API for that
  - but, more code/more difficult –
  - this example uses an API that does exactly one (1) thing.
- There are a TON of other (free)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

  Spotify, LastFM, Twitter, bol.com, etc



https://zipcodeapi.com/

Register your ‘application’



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;
```

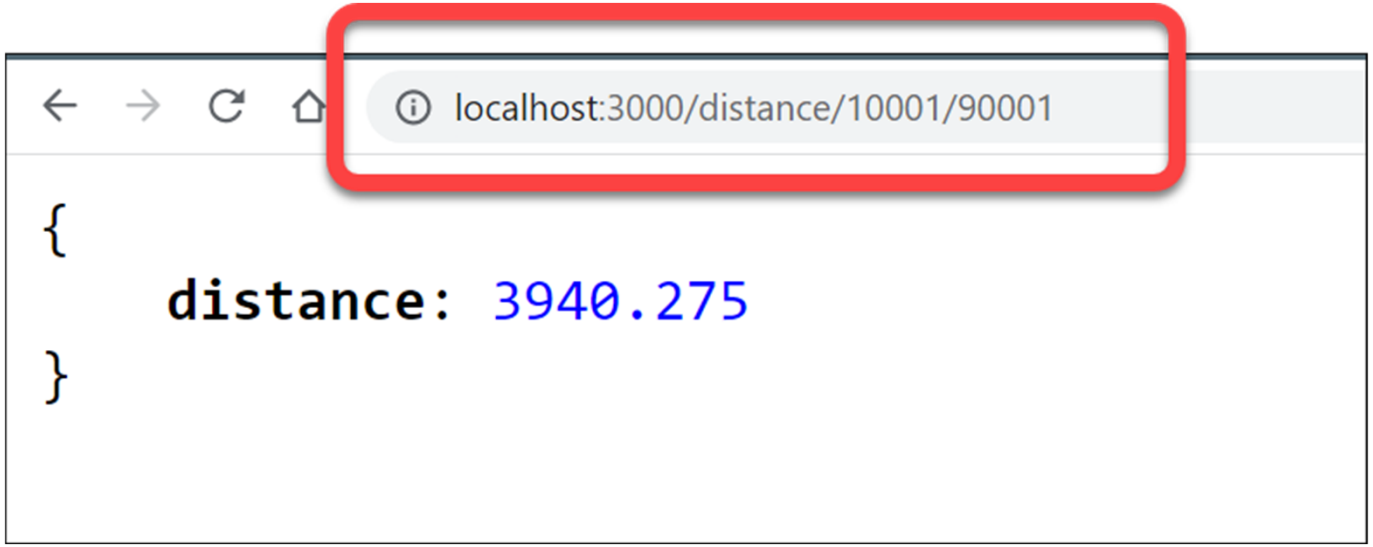
Test your service

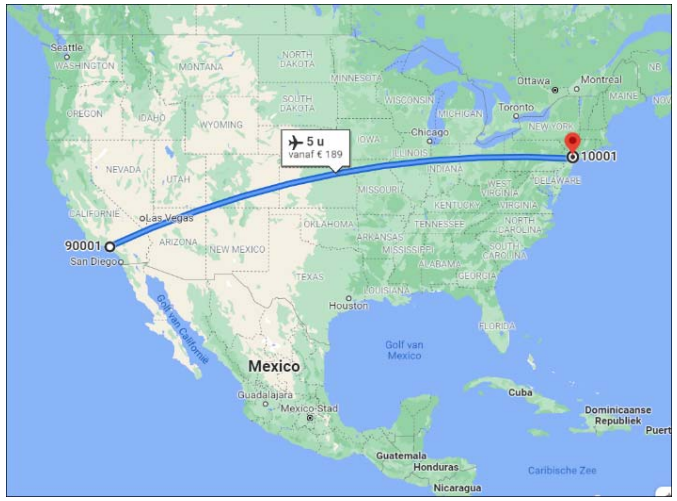
```
> api-request-app@1.0.0 start C:\Users\Gebruiker\Desktop\request-app
> node server.js

Listening on port http://localhost:3000
```

localhost:3000/about

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



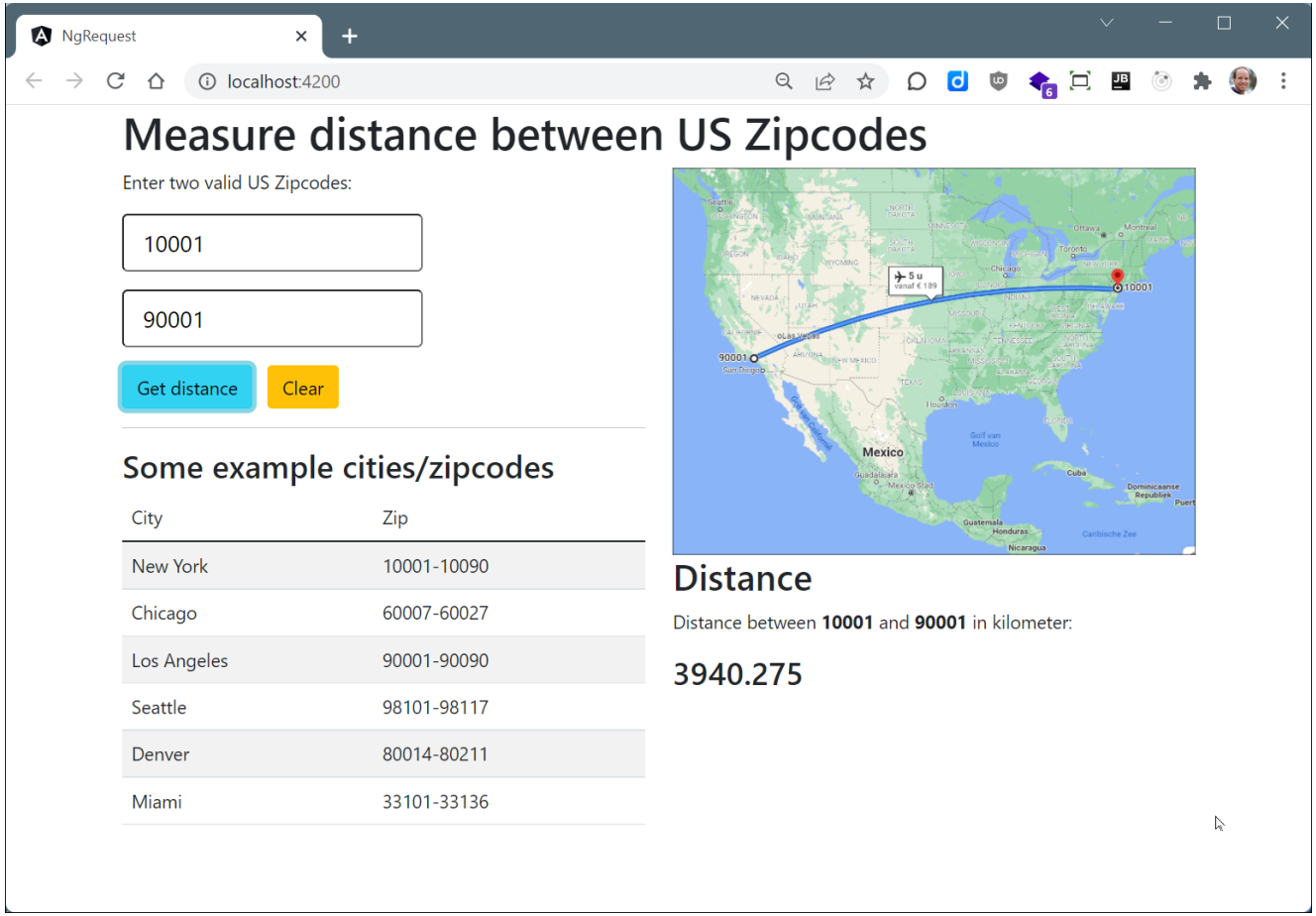


Very simple. Calculate distance (in km) between New York (10001) and Los Angeles (90001)

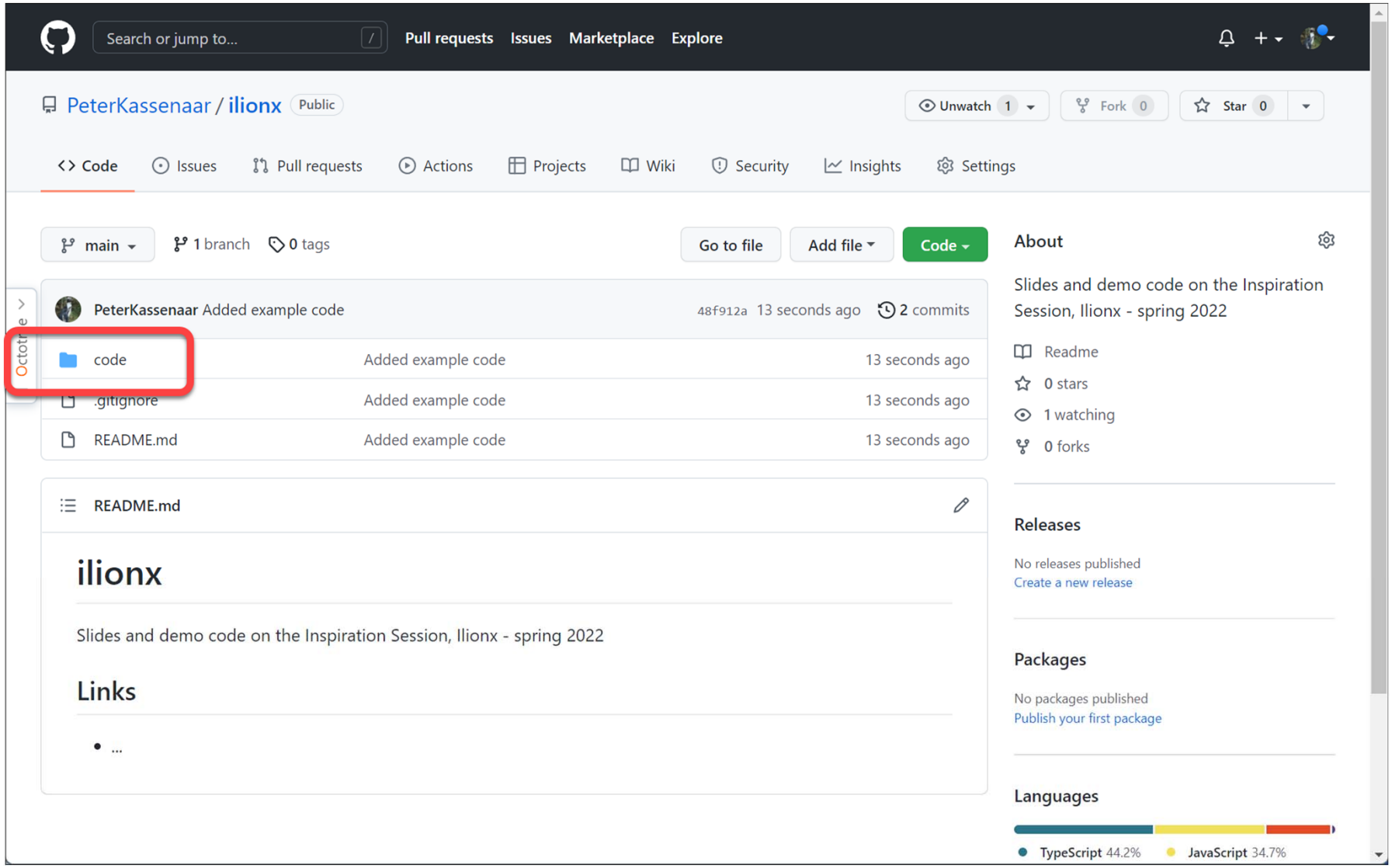
Build your Frontend

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

- Start server + start frontend



Example code



Questions?