



SPRING BOOT VS. .NET CORE

It's all about the APIs



AGENDA

- Prerequisites
- Dependencies management
- Creating a new api
- Create CRUD Controller
- Validation
- Error handling
- Database access
- Call another API
- Scheduled Jobs
- Testing controllers
- Memory footprint
- Swagger
- Docker
- Deploy to heroku

.Net Core WebApi vs Spring Boot

This is a presentation about the 2 technologies and it's purpose is to highlight the similarities and differences between them.

The demo project is a Todos API with basic endpoints and a postgresql database. It explains the controller, model, service, validation, error handling, api to api call, scheduled jobs, testing, memory footprint and docker files.

Presentation can be found here: [Open slides](#)

Bitly short url for all files: <http://bit.ly/core-vs-spring>

It also contains 2 projects with the same features for a better comparison.

Here are the projects running live:

- dotnet core 2/3 REST webapi [.net core webapi link](#)
- spring boot 2 REST api [spring boot api link](#)

The keynote presentation link is here: [open me](#)

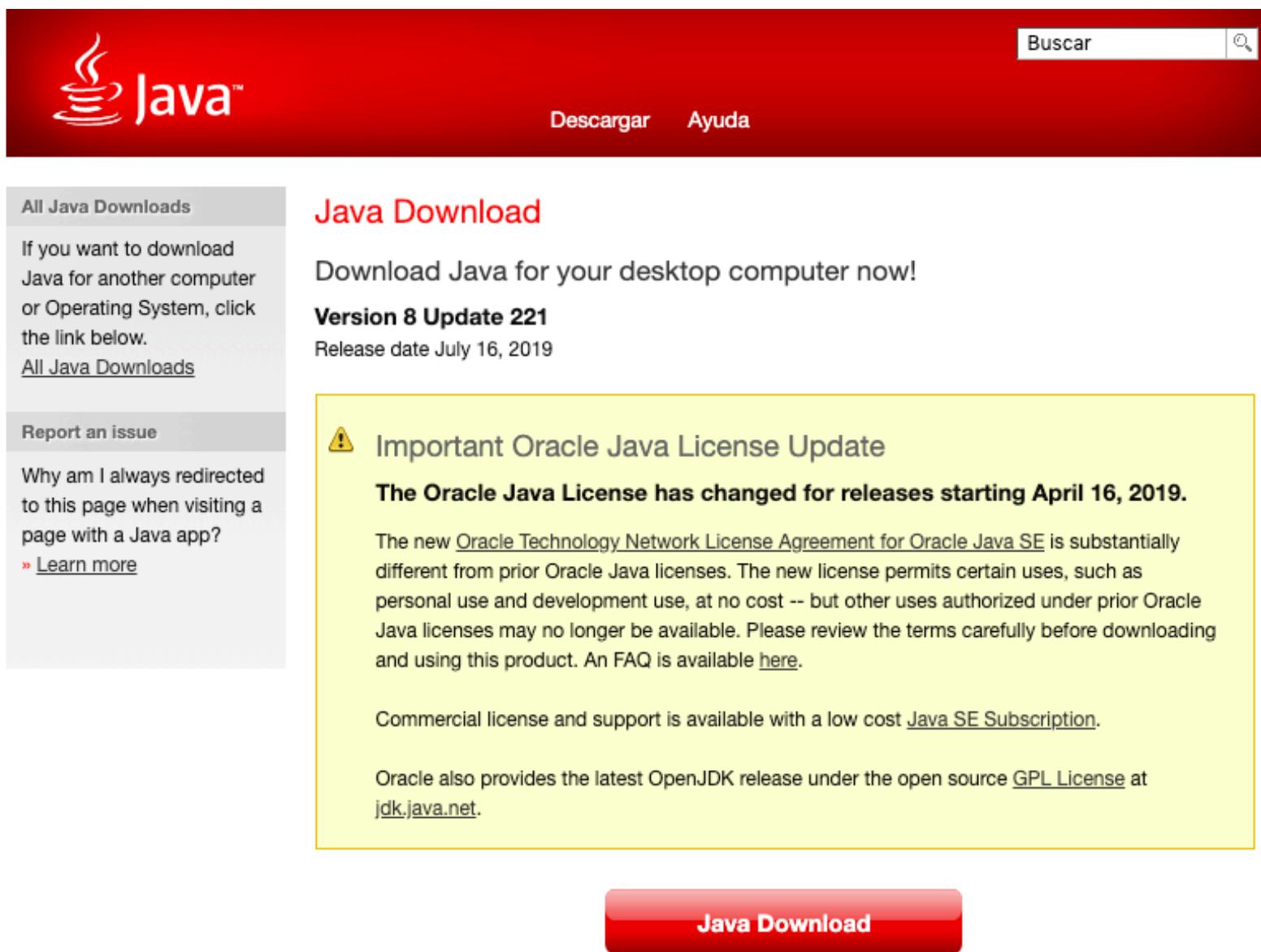
In each project there is a Dockerfile and a Docker-compose file so that it can be executed locally. The docker file is expecting a postgres installation to run on your localhost port 5432. For the docker-compose there is no other requirement than the docker to be present.

NJoy

PREREQUISITES

Install Java

<https://www.java.com/ES/download/>



The screenshot shows the Java Download page. At the top, there's a red header with the Java logo and navigation links for 'Descargar' and 'Ayuda'. Below the header, a sidebar on the left contains links for 'All Java Downloads', 'Report an issue', and 'Important Oracle Java License Update'. The main content area features a yellow box titled 'Important Oracle Java License Update' with text about the license change on April 16, 2019. It also mentions OpenJDK and Java SE Subscriptions. A large red 'Java Download' button is at the bottom.

Install .Net Core

<https://dotnet.microsoft.com/download>



The screenshot shows the .NET Download page. It has a purple header with the text 'Download .NET' and a subtext 'Downloads for .NET Framework and .NET Core, including ASP.NET and ASP.NET Core'. Below the header, there's a callout box with a question mark icon and text about starting with the 'Hello World in 10 minutes tutorial'. At the bottom, there are download links for 'Windows', 'Linux', 'macOS' (which is underlined), and 'Docker'.



.NET Core 3.0

.NET Core is a cross-platform version of .NET for building websites, services, and console apps.

- Build Apps ⓘ [Download .NET Core SDK](#)
- Advanced ⓘ [All .NET Core downloads...](#)

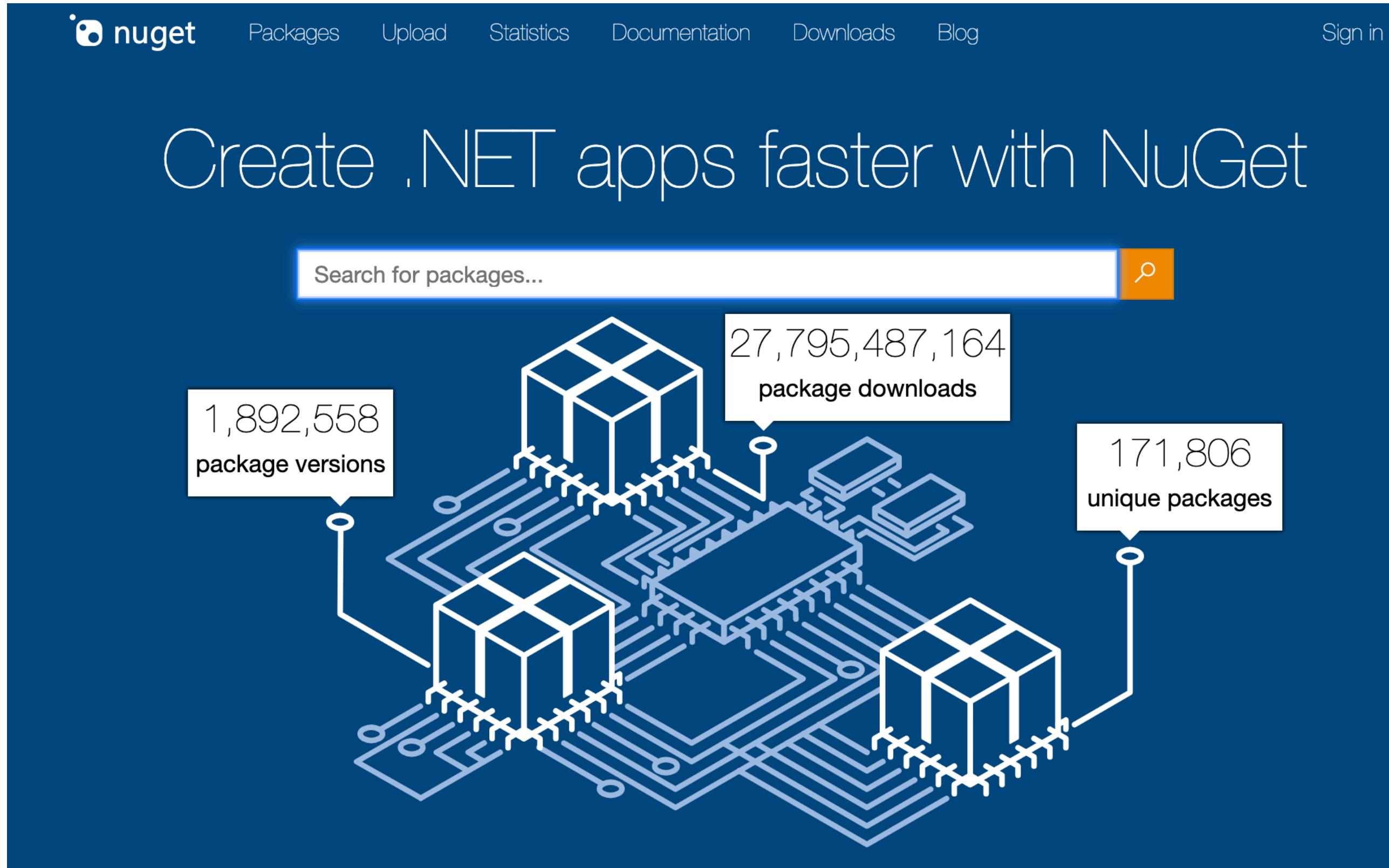
DEPENDENCIES MANAGEMENT



Apache / Maven / Welcome to Apache Maven



[Download](#) | [Get Sources](#) | Last Published: 2019-10-08

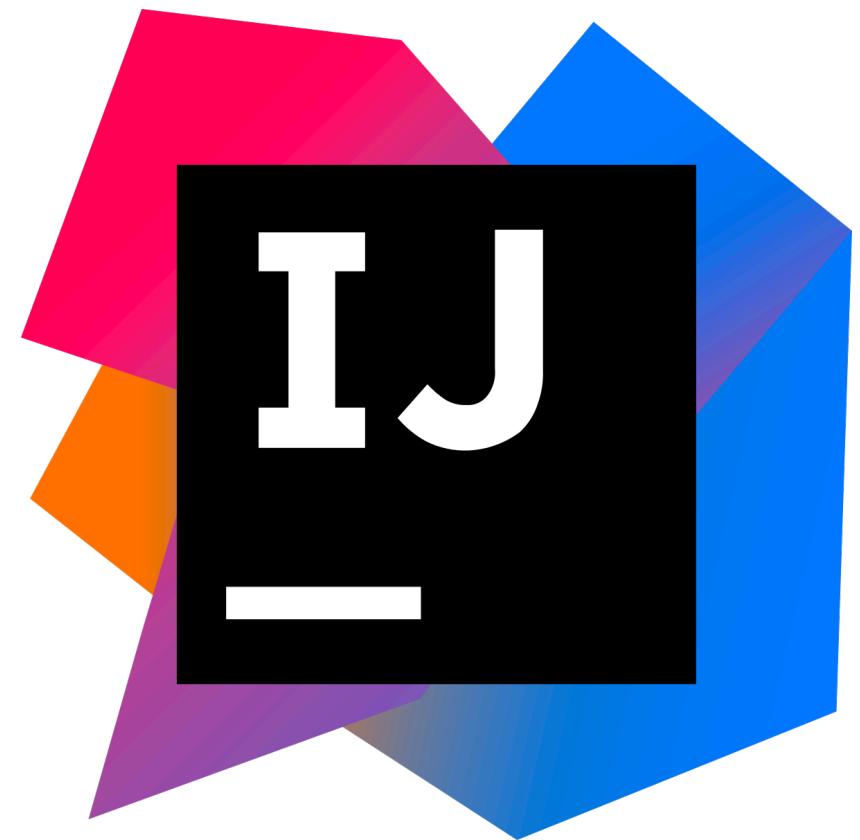


The screenshot shows the NuGet homepage with a dark blue background. At the top, there's a navigation bar with links for "nuget", "Packages", "Upload", "Statistics", "Documentation", "Downloads", "Blog", and "Sign in". Below the navigation, the text "Create .NET apps faster with NuGet" is displayed. A search bar with the placeholder "Search for packages..." and a magnifying glass icon is positioned above a central graphic. The graphic features a blue circuit board with three white 3D cube icons representing packages. Callout boxes provide statistics: "1,892,558 package versions", "27,795,487,164 package downloads", and "171,806 unique packages".

INTEGRATED DEVELOPMENT ENVIRONMENT: DOT NET



INTEGRATED DEVELOPMENT ENVIRONMENT: JAVA



Visual Studio Code

CREATE NEW API - .NET CORE

Create a new project

.net core Language Platform Project type

Recent project templates

Filtering by: Web, C# Clear filter

MFC App C++ ASP.NET Core Web Application
Project templates for creating ASP.NET Core applications for Windows, Linux and macOS using .NET Core or .NET Framework. Create Razor Pages, MVC, Web API, and Single Page (SPA) Applications.
C# Windows Linux macOS Web

NUnit Test Project (.NET Core)
A project that contains NUnit tests that can run on .NET Core on Windows, Linux and MacOS.
C# Linux macOS Windows Desktop Test Web

Web Driver Test for Edge (.NET Core)
A project that contains unit tests that can automate UI testing of web sites within Edge browser (using Microsoft WebDriver).
C# Windows Web Test

Not finding what you're looking for?
Install more tools and features

Next

CREATE NEW API - .NET CORE

Create a new ASP.NET Core Web Application

.NET Core ASP.NET Core 2.1

API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

Web Application
A project template for creating an ASP.NET Core application with example ASP.NET Core Razor Pages content.

Web Application (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

Razor Class Library
A project template for creating a Razor class library.

Angular
A project template for creating an ASP.NET Core application with Angular.

React.js
A project template for creating an ASP.NET Core application with React.js.

[Get additional project templates](#)

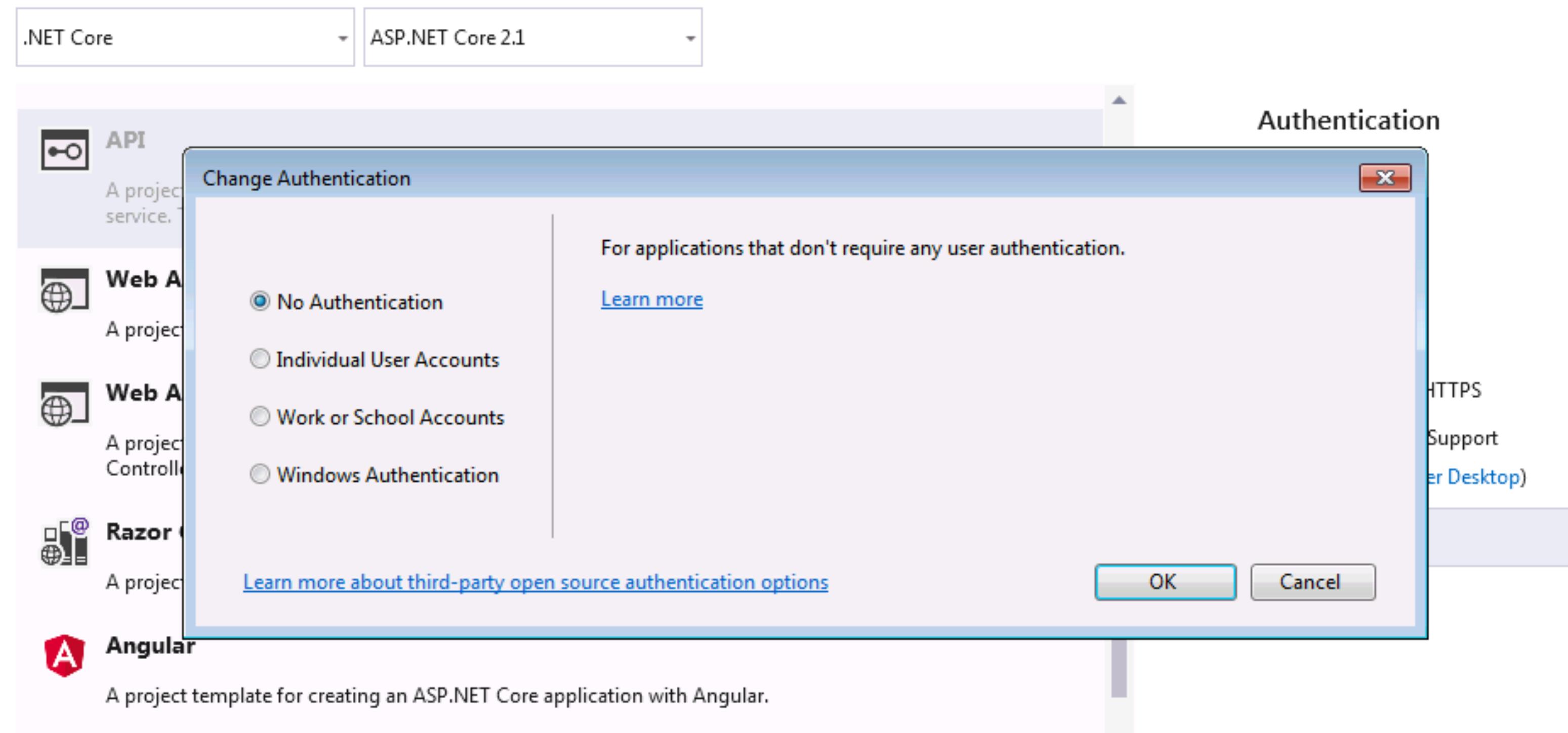
Authentication
No Authentication
[Change](#)

Advanced
 Configure for HTTPS
 Enable Docker Support
(Requires [Docker Desktop](#))
Linux

Author: Microsoft
Source: SDK 2.1.700

[Back](#) [Create](#)

CREATE NEW API - .NET CORE



CREATE NEW API - .NET CORE

```
➔ ~ dotnet new
Usage: new [options]

Options:
  -h, --help          Displays help for this command.
  -l, --list           Lists templates containing the specified name. If no name is specified, lists all templates.
  -n, --name            The name for the output being created. If no name is specified, the name of the current directory is used.
  -o, --output          Location to place the generated output.
  -i, --install         Installs a source or a template pack.
  -u, --uninstall       Uninstalls a source or a template pack.
  --nuget-source        Specifies a NuGet source to use during install.
  --type                Filters templates based on available types. Predefined values are "project", "item" or "other".
  --dry-run              Displays a summary of what would happen if the given command line were run if it would result in a template creation.
  --force                Forces content to be generated even if it would change existing files.
  -lang, --language      Filters templates based on language and specifies the language of the template to create.
  --update-check         Check the currently installed template packs for updates.
  --update-apply         Check the currently installed template packs for update, and install the updates.
```

Templates	Short Name	Language	Tags
Console Application	console	[C#], F#, VB	Common/Console
Class library	classlib	[C#], F#, VB	Common/Library
WPF Application	wpf	[C#]	Common/WPF
WPF Class library	wpflib	[C#]	Common/WPF
WPF Custom Control Library	wpfcustomcontrollib	[C#]	Common/WPF
WPF User Control Library	wpfusercontrollib	[C#]	Common/WPF
Windows Forms (WinForms) Application	winforms	[C#]	Common/WinForms
Windows Forms (WinForms) Class library	winformslib	[C#]	Common/WinForms
Worker Service	worker	[C#]	Common/Worker/Web
Unit Test Project	mstest	[C#], F#, VB	Test/MSTest
NUnit 3 Test Project	nunit	[C#], F#, VB	Test/NUnit
NUnit 3 Test Item	nunit-test	[C#], F#, VB	Test/NUnit
xUnit Test Project	xunit	[C#], F#, VB	Test/xUnit
Razor Component	razorcomponent	[C#]	Web/ASP.NET
Razor Page	page	[C#]	Web/ASP.NET

CREATE NEW API - SPRING BOOT: [START.SPRING.IO](https://start.spring.io)

The screenshot shows the Spring Initializr web application at start.spring.io. The configuration is set up for a **Maven Project** in **Java** using **Spring Boot 2.1.9**. The **Project Metadata** section includes a Group of `com.todos` and an Artifact of `api`. In the **Dependencies** section, the user has searched for "Web, Security, JPA, Actuator, Devtools..." and selected the **Spring Web** dependency, which is described as building a web application using Spring MVC and Apache Tomcat. The interface also includes links for **Generate**, **Explore**, and **Share**.

Project: Maven Project

Language: Java

Spring Boot: 2.1.9

Project Metadata:

- Group: com.todos
- Artifact: api

Dependencies:

- Search dependencies to add: Web, Security, JPA, Actuator, Devtools...
- Selected dependencies:

 - Spring Web**: Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Generate - ⌘ + ↩ Explore - Ctrl + Space Share...

DEPENDENCIES

Selected dependencies

Spring Web

Build web, including RESTful, applications using Spring MVC.
Uses Apache Tomcat as the default embedded container.



Spring Data JPA

Persist data in SQL stores with Java Persistence API using
Spring Data and Hibernate.



Spring Boot DevTools

Provides fast application restarts, LiveReload, and
configurations for enhanced development experience.



PostgreSQL Driver

A JDBC and R2DBC driver that allows Java programs to
connect to a PostgreSQL database using standard, database
independent Java code.



Lombok

Java annotation library which helps to reduce boilerplate
code.



TODO MODEL C# | JAVA

.....

The image shows a developer's workspace with two code editors side-by-side, illustrating the implementation of a TODO model in both C# and Java.

Left Editor (C#):

```
using System;
namespace api.Model
{
    public class Todo
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public bool IsCompleted { get; set; }
    }
}
```

Right Editor (Java):

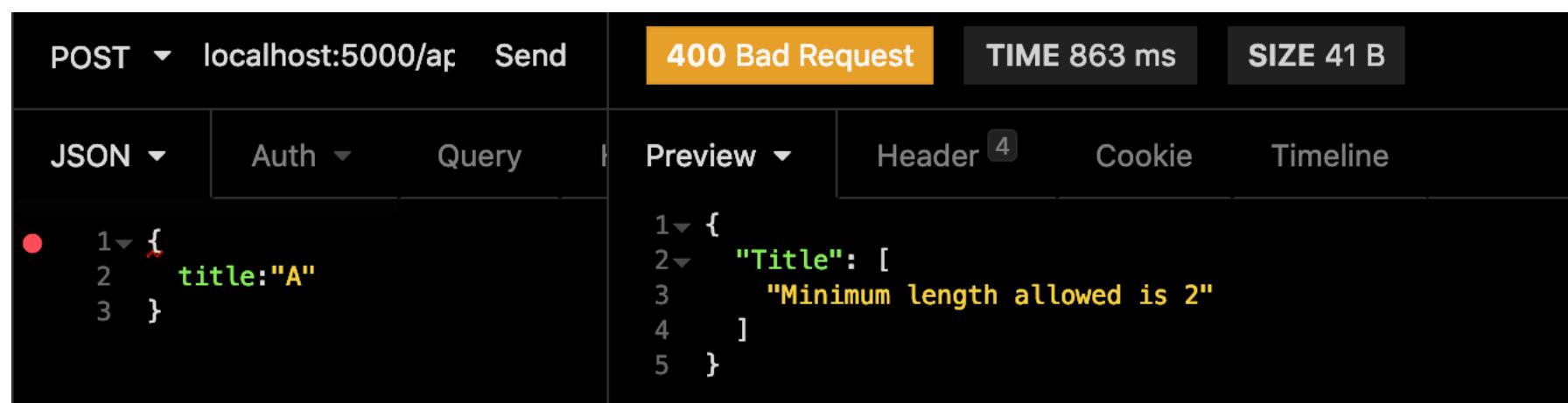
```
package com.todos.api.models;
import lombok.*;
@Builder
@AllArgsConstructor
public class Todo {
    private int Id;
    private String title;
    private boolean isCompleted;
}
```

The Java code uses Lombok annotations (`@Getter`, `@Setter`, `@Builder`, and `@AllArgsConstructor`) to generate boilerplate code for getters, setters, and a constructor. The code structure follows a standard Spring Boot directory layout with `src/main/java` containing the `com.todos.api.controllers` and `models` packages.

DOMAIN VALIDATION C#

```
public class Todo
{
    [Required]
    10 references
    public int Id { get; set; }
    [Required]
    [MinLength(2, ErrorMessage = "Minimum length allowed is 2")]
    8 references
    public string Title { get; set; }
```

```
// POST api/todos
[HttpPost]
0 references
public ActionResult Post([FromBody]Todo todo)
{
    if (!ModelState.IsValid)
        return BadRequest(ModelState);
    _service.Create(todo);
    return Created("", todo);
}
```



Built-in attributes

Here are some of the built-in validation attributes:

- `[CreditCard]` : Validates that the property has a credit card format.
- `[Compare]` : Validates that two properties in a model match.
- `[EmailAddress]` : Validates that the property has an email format.
- `[Phone]` : Validates that the property has a telephone number format.
- `[Range]` : Validates that the property value falls within a specified range.
- `[RegularExpression]` : Validates that the property value matches a specified regular expression.
- `[Required]` : Validates that the field is not null. See [\[Required\] attribute](#) for details about this attribute's behavior.
- `[StringLength]` : Validates that a string property value doesn't exceed a specified length limit.
- `[Url]` : Validates that the property has a URL format.
- `[Remote]` : Validates input on the client by calling an action method on the server. See [\[Remote\] attribute](#) for details about this attribute's behavior.

```
public class ValidEmailDomainAttribute : ValidationAttribute
{
    private readonly string allowedDomain;

    public ValidEmailDomainAttribute(string allowedDomain)
    {
        this.allowedDomain = allowedDomain;
    }

    public override bool IsValid(object value)
    {
        string[] strings = value.ToString().Split('@');
        return strings[1].ToUpper() == allowedDomain.ToUpper();
    }
}
```

DOMAIN VALIDATION JAVA

JSR 380 is a specification of the Java API for bean validation, part of JavaEE and JavaSE, which ensures that the properties of a bean meet specific criteria, using annotations such as `@NotNull`, `@Min`, and `@Max`.

```
public class Todo {  
    @Id @GeneratedValue  
    private int Id;  
    @NotNull  
    @Min(value = 2,message = "Minimum length allowed is 2")  
    private String title;
```

```
@PostMapping(value = "/api/todos")  
public ResponseEntity<?> post(@Valid @RequestBody Todo todo){  
    _service.create(todo);  
    return ResponseEntity.created( location: null).body(todo);  
}
```

```
{  
    "timestamp": "2019-11-20T15:04:53.583+0000",  
    "status": 400,  
    "error": "Bad Request",  
    "errors": [  
        {  
            "codes": [ ],  
            "arguments": [ ],  
            "defaultMessage": "Minimum length allowed is 2",  
            "objectName": "todo",  
            "field": "title",  
            "rejectedValue": "s",  
            "bindingFailure": false,  
            "code": "Min"  
        }  
    ],  
    "message": "Validation failed for object='todo'. Error count: 1",  
}
```

Per the JSR 380 specification, the `validation-api` dependency contains the standard validation APIs.

Hibernate Validator is the reference implementation of the validation API.

All of the annotations used in the example are standard JSR annotations:

- `@NotNull` – validates that the annotated property value is not `null`
- `@AssertTrue` – validates that the annotated property value is `true`
- `@Size` – validates that the annotated property value has a size between the attributes `min` and `max`, can be applied to `String`, `Collection`, `Map`, and array properties
- `@Min` – validates that the annotated property has a value no smaller than the `value` attribute
- `@Max` – validates that the annotated property has a value no larger than the `value` attribute
- `@Email` – validates that the annotated property is a valid email address

Some annotations accept additional attributes, but the `message` attribute is common to all of them. This is the message that will usually be rendered when the value of the respective property fails validation.

Some additional annotations that can be found in the JSR are:

- `@NotEmpty` – validates that the property is not null or empty; can be applied to `String`, `Collection`, `Map` or `Array` values
- `@NotBlank` – can be applied only to text values and validated that the property is not null or whitespace
- `@Positive` and `@PositiveOrZero` – apply to numeric values and validate that they are strictly positive, or positive including 0
- `@Negative` and `@NegativeOrZero` – apply to numeric values and validate that they are strictly negative, or negative including 0
- `@Past` and `@PastOrPresent` – validate that a date value is in the past or the past including the present; can be applied to date types including those added in Java 8
- `@Future` and `@FutureOrPresent` – validates that a date value is in the future, or in the future including the present

CUSTOM DOMAIN VALIDATION JAVA

4. The New Annotation

Let's create a new `@interface` to define our annotation:

```
1  @Documented
2  @Constraint(validatedBy = ContactNumberValidator.class)
3  @Target( { ElementType.METHOD, ElementType.FIELD })
4  @Retention(RetentionPolicy.RUNTIME)
5  public @interface ContactNumberConstraint {
6      String message() default "Invalid phone number";
7      Class<?>[] groups() default {};
8      Class<? extends Payload>[] payload() default {};
9  }
```

5. Creating a Validator

Let's now create a validator class that enforces rules of our validation:

```
1  public class ContactNumberValidator implements
2      ConstraintValidator<ContactNumberConstraint, String> {
3
4      @Override
5      public void initialize(ContactNumberConstraint contactNumber) {
6
7      }
8
8      @Override
9      public boolean isValid(String contactField,
10          ConstraintValidatorContext cxt) {
11          return contactField != null && contactField.matches("[0-9]+")
12              && (contactField.length() > 8) && (contactField.length() < 14);
13      }
14  }
```

5. Programmatic Validation

Some frameworks – such as Spring – have simple ways of triggering the validation process by just using annotations. This is mainly so that we don't have to interact with the programmatic validation API.

Let's now go the manual route and set things up programmatically:

```
1  ValidatorFactory factory = Validation.buildDefaultValidatorFactory();
2  Validator validator = factory.getValidator();
```

```
1  Set<ConstraintViolation<User>> violations = validator.validate(user);
```

By iterating over the violations, we can get all the violation messages by using the `getMessage` method.

```
1  for (ConstraintViolation<User> violation : violations) {
2      log.error(violation.getMessage());
3  }
```

TODO SERVICE C# | JAVA

```
namespace api.Servcies
{
    public class TodosService
    {
        List<Todo> _todos = new List<Todo> {
            new Todo{Id=1, Title="First todo!", IsCompleted = false};

        public List<Todo> GetAll() => _todos;
        public Todo Get(int id) => _todos.FirstOrDefault(x => x.Id == id);
        public void Create(Todo todo) {
            todo.Id = _todos.Count + 1;
            _todos.Add(todo);
        }

        public void Update(int id,Todo todo)
        {
            var td = _todos.FirstOrDefault(x => x.Id == id);
            if (td != null)
            {
                td.Title = todo.Title;
                td.IsCompleted = todo.IsCompleted;
            }
        }
        public void Delete(int id) =>
            _todos.Remove(_todos.Single(x => x.Id == id));
    }
}
```

```
@Service
public class TodosService {
    private List<Todo> _todos = new ArrayList<>();

    {
        _todos.add(new Todo( Id: 1, title: "First todo", isCompleted: false));
    };

    public List<Todo> GetAll(){ return _todos;}
    public Todo Get(int id){
        return _todos.stream().filter(x->x.getId() == id).findFirst().get();

    }

    public Todo create(Todo todo) {
        todo.setId(_todos.size() + 1);
        _todos.add(todo);
        return todo;
    }

    public Todo update(int id, Todo todo) {

        Todo td = _todos.stream().filter(x-> x.getId() == id)
            .findFirst().get();
        if (td != null){
            td.setCompleted(todo.isCompleted());
            td.setTitle(todo.getTitle());
        }
        return td;
    }

    public void delete(int id){
        _todos.remove(
            _todos.stream().filter(x-> x.getId() == id)
            .findFirst().get());
    }
}
```

TODOS CONTROLLER C# | JAVA

```
[Route("api/[controller]")]
public class TodosController : Controller
{
    public TodosService _service { get; }

    public TodosController(TodosService service)
        => _service = service;

    // GET: api/todos
    [HttpGet]
    public IEnumerable<Todo> Get() => _service.GetAll();

    // GET api/todos/5
    [HttpGet("{id}")]
    public ActionResult<Todo> Get(int id) => Ok(_service.Get(id));

    // POST api/todos
    [HttpPost]
    public ActionResult Post([FromBody]Todo todo)
    {
        _service.Create(todo);
        return Created("", todo);
    }

    // PUT api/todos/5
    [HttpPut("{id}")]
    public ActionResult Put(int id, [FromBody]Todo todo)
    {
        _service.Update(id, todo);
        return NoContent();
    }

    // DELETE api/todos/5
    [HttpDelete("{id}")]
    public ActionResult Delete(int id)
    {
        _service.Delete(id);
        return NoContent();
    }
}
```

```
@RestController
//@RequestMapping()
public class TodosController {

    @Autowired
    TodosService _service;

    @GetMapping(value = "/api/todos")
    public ResponseEntity< List<Todo>> get(){
        return ResponseEntity.ok( _service.GetAll());
    }

    @GetMapping(value = "/api/todos/{id}")
    public ResponseEntity<?> get(@PathVariable Integer id){
        return ResponseEntity.ok(_service.Get(id));
    }

    @PostMapping(value = "/api/todos")
    public ResponseEntity<?> post(@RequestBody Todo todo){
        _service.create(todo);
        return ResponseEntity.created( location: null).body(todo);
    }

    @PutMapping(value = "/api/todos/{id}")
    public ResponseEntity<Todo> put(@PathVariable Integer id, @RequestBody Todo todo){
        Todo td = _service.update(id, todo);
        return ResponseEntity.ok(td);
    }

    @DeleteMapping(value = "/api/todos/{id}")
    public ResponseEntity<?> delete(@PathVariable Integer id){
        _service.delete(id);
        return ResponseEntity.accepted().build();
    }
}
```

TODOS ERROR HANDLING: C#

```
 1 usage
public class Startup
{
    public Startup(IConfiguration configuration){...}
    1 usage
    public IConfiguration Configuration { get; }
    // This method gets called by the runtime. Use this method to add services to the container.
    public void ConfigureServices(IServiceCollection services){...}
    // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
    public void Configure(IApplicationBuilder app, IHostingEnvironment env)
    {

        if (env.IsDevelopment())
            app.UseDeveloperExceptionPage();

        app.ConfigureExceptionHandler();
    }
}
```

```
 1 usage
public static void ConfigureExceptionHandler(this IApplicationBuilder app)
{
    app.UseExceptionHandler(appError =>
    {
        appError.Run( handler: async context =>
        {
            context.Response.StatusCode = (int) HttpStatusCode.InternalServerError;
            context.Response.ContentType = "application/json";

            var contextFeature = context.Features.Get<IExceptionHandlerFeature>();
            if (contextFeature != null)
            {
                Trace.WriteLine( message: $"Something went wrong: {contextFeature.Error}");

                await context.Response.WriteAsync(
                    text: JsonConvert.SerializeObject(new {
                        StatusCode = (int)context.Response.StatusCode,
                        Message = "Internal Server Error."
                    })
                );
            }
        });
    });
}
```

TODOS ERROR HANDLING: JAVA

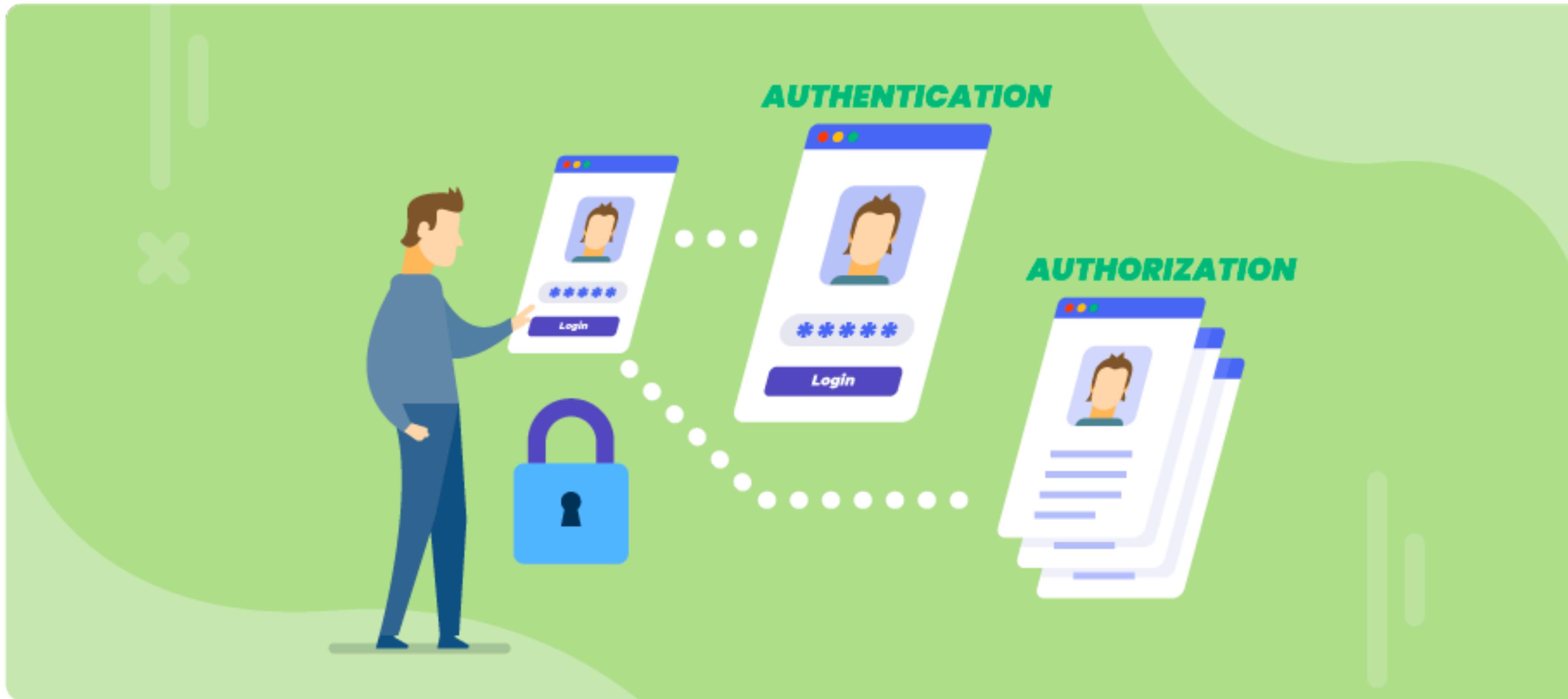
2. Solution 1 – The Controller level *@ExceptionHandler*

The first solution works at the *@Controller* level – we will define a method to handle exceptions, and annotate that with *@ExceptionHandler*:

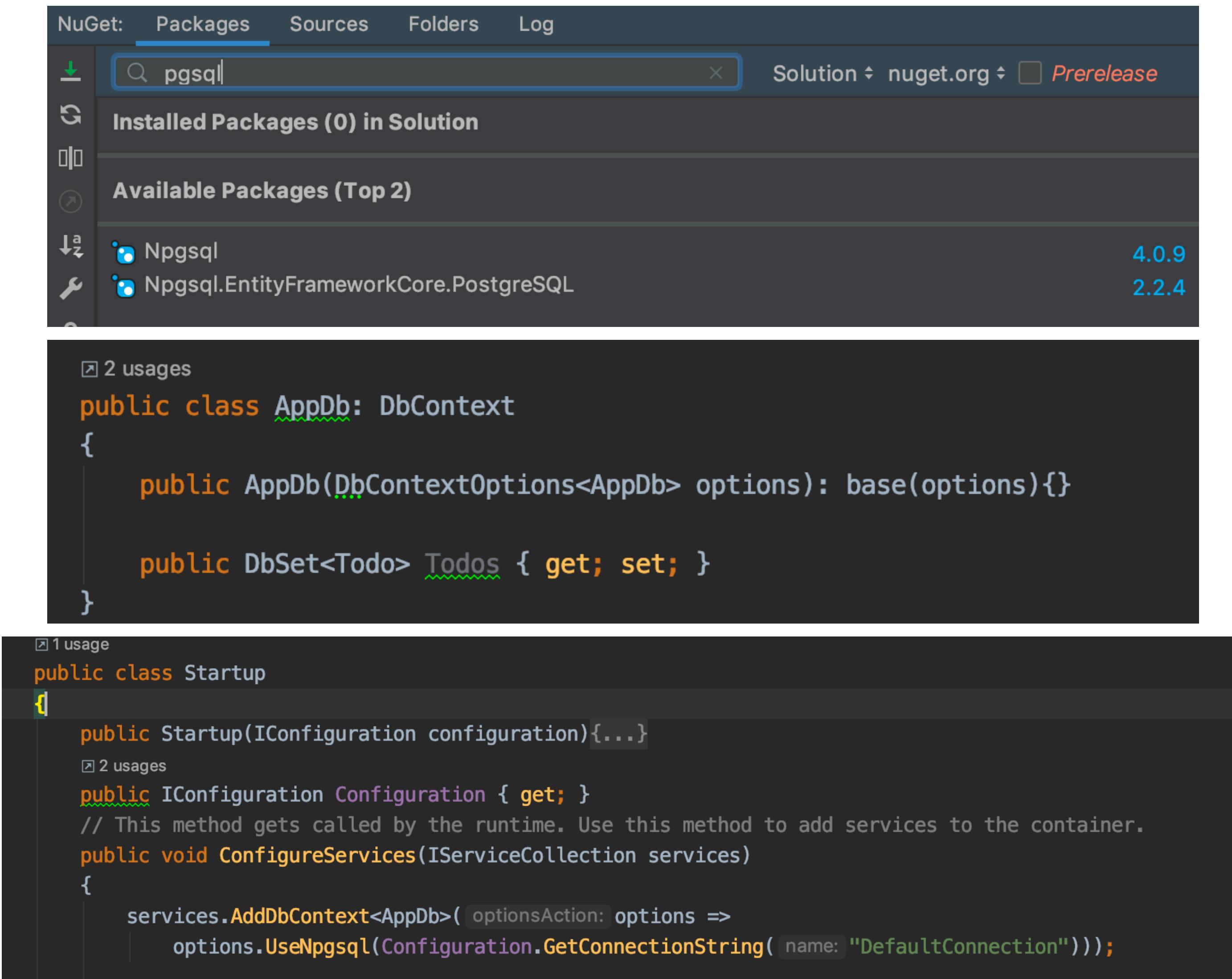
```
1 public class FooController{  
2  
3     //...  
4     @ExceptionHandler({ CustomException1.class, CustomException2.class })  
5     public void handleException() {  
6         //  
7     }  
8 }
```

```
@Data  
@Builder  
class JSONException{  
    private String message;  
}  
  
@RestControllerAdvice  
public class TodosControllerAdvice {  
  
    @ExceptionHandler(TodoException.class)  
    @ResponseStatus(HttpStatus.INTERNAL_SERVER_ERROR)  
    public ResponseEntity<?> handleNonExistingHero(TodoException exp) {  
        return new ResponseEntity<?>( new JSONException( exp.getMessage()), HttpStatus.INTERNAL_SERVER_ERROR);  
    }  
}
```

AUTHENTICATION & AUTHORIZATION



DATABASE ACCESS C#



NuGet: Packages Sources Folders Log

pgsql

Solution nuget.org Prerelease

Installed Packages (0) in Solution

Available Packages (Top 2)

Package	Version
Npgsql	4.0.9
Npgsql.EntityFrameworkCore.PostgreSQL	2.2.4

2 usages

```
public class AppDb: DbContext
{
    public AppDb(DbContextOptions<AppDb> options): base(options){}
    public DbSet<Todo> Todos { get; set; }
}
```

1 usage

```
public class Startup
{
    public Startup(IConfiguration configuration){...}
    public IConfiguration Configuration { get; }
    // This method gets called by the runtime. Use this method to add services to the container.
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddDbContext<AppDb>( optionsAction: options =>
            options.UseNpgsql(Configuration.GetConnectionString( name: "DefaultConnection")));
    }
}
```

DATABASE ACCESS JAVA

pom.xml

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <scope>runtime</scope>
</dependency>
```

application.properties

```
spring.datasource.url= jdbc:postgresql://localhost:5432/todos_java
spring.datasource.username=postgres
spring.datasource.password=postgres
spring.jpa.hibernate.ddl-auto=update
spring.datasource.driver-class-name=org.postgresql.Driver

spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
```

```
1 package com.todos.api.models;
2
3
4 import lombok.*;
5 import javax.persistence.*;
6
7 @Getter
8 @Setter
9 @AllArgsConstructor
10 @NoArgsConstructor
11 @Entity
12 public class Todo {
13     @Id @GeneratedValue
14     private int id;
15     private String title;
16     private boolean isCompleted;
17 }
18 |
```

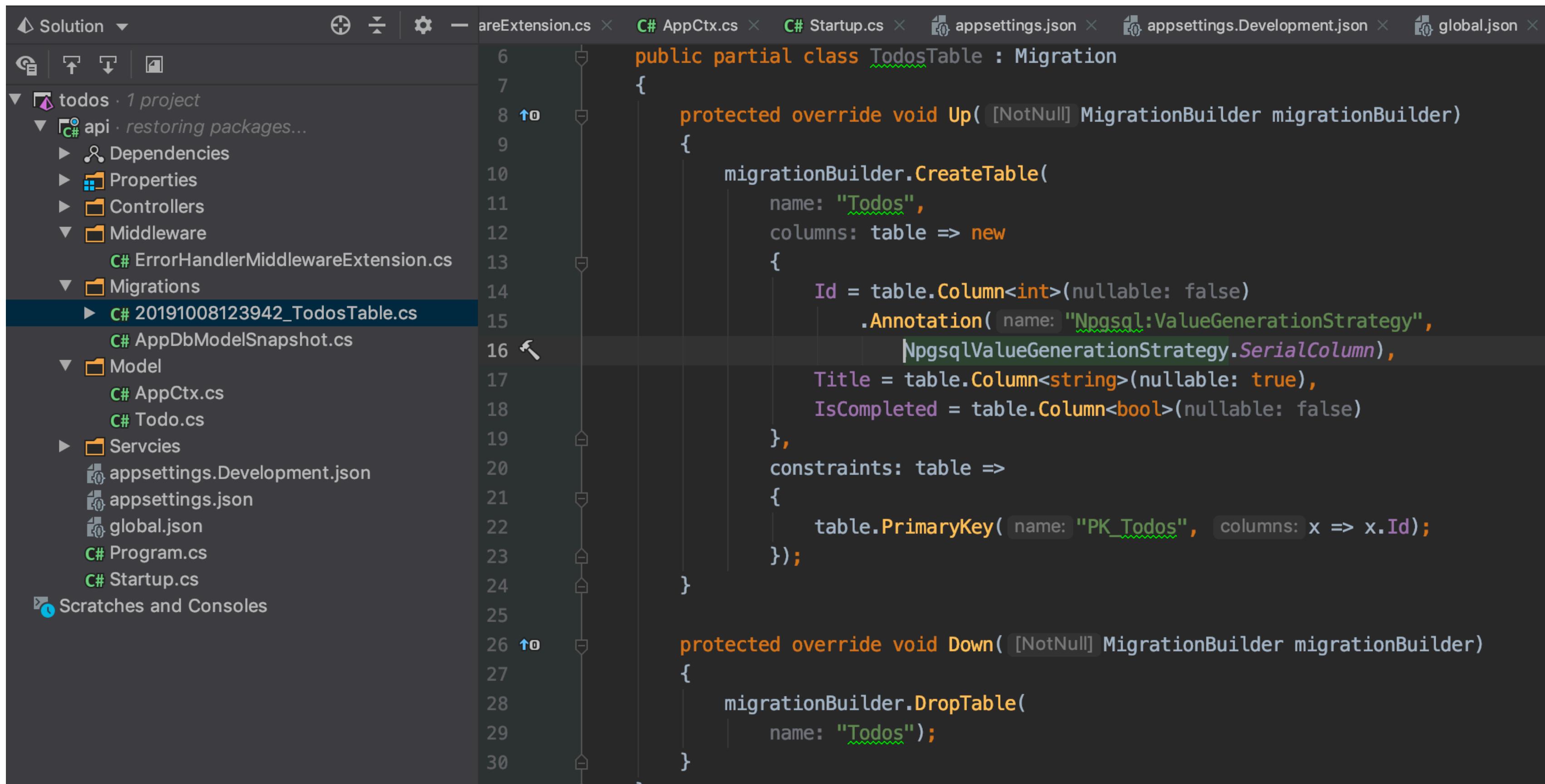
```
package com.todos.api.repositories;

import com.todos.api.models.Todo;
import org.springframework.data.jpa.repository.JpaRepository;

public interface TodosRepository extends JpaRepository<Todo, Integer> { }
```

CREATE MIGRATIONS: C#

→ *dotnet ef migrations add TodosTable*



The screenshot shows the Visual Studio IDE interface with the 'Todos' project selected in the Solution Explorer. The 'Migrations' folder contains a file named '20191008123942_TodosTable.cs'. The code in this file defines a migration for creating a 'Todos' table:

```
public partial class TodosTable : Migration
{
    protected override void Up([NotNull] MigrationBuilder migrationBuilder)
    {
        migrationBuilder.CreateTable(
            name: "Todos",
            columns: table => new
            {
                Id = table.Column<int>(nullable: false)
                    .Annotation(name: "Npgsql:ValueGenerationStrategy", value: "NpgsqlValueGenerationStrategy.SerialColumn"),
                Title = table.Column<string>(nullable: true),
                IsCompleted = table.Column<bool>(nullable: false)
            },
            constraints: table =>
            {
                table.PrimaryKey(name: "PK_Todos", columns: x => x.Id);
            });
    }

    protected override void Down([NotNull] MigrationBuilder migrationBuilder)
    {
        migrationBuilder.DropTable(
            name: "Todos");
    }
}
```

→ *dotnet ef database update*

AUTOUPDATE DATABASE: C#

Inside the Startup class

```
// update database schema
using (var serviceScope = app.ApplicationServices.GetRequiredService<IServiceScopeFactory>().CreateScope())
{
    if (serviceScope.ServiceProvider.GetService<AppDb>() == null) return;
    var ctx = serviceScope.ServiceProvider.GetService<AppDb>();
    new DatabaseFacade(ctx).Migrate();
}
```

DATABASE MIGRATIONS: JAVA



**Version control for your database.
Robust schema evolution across all your environments.
With ease, pleasure and plain SQL.**



The leading open source tool for database change and deployment management.

Which Option is Right for You?

USE THE DATABASE: TODOSERVICE C# | JAVA

```
8     public class TodosService
9     {
10         private readonly AppDb _db;
11
12         public TodosService(){}
13         public TodosService(AppDb db) => _db = db;
14
15         public virtual List<Todo> GetAll() =>
16             _db.Todos.OrderByDescending(x=>x.Id).ToList();
17
18         public Todo Get(int id) =>
19             _db.Todos.Single(x => x.Id == id);
20
21         public void Create(Todo todo) {
22             _db.Todos.Add(todo);
23             _db.SaveChanges();
24         }
25
26         public Todo Update(int id,Todo todo)
27         {
28             var td = _db.Todos.FirstOrDefault(x => x.Id == id);
29             if (td != null)
30             {
31                 td.Title = todo.Title;
32                 td.IsCompleted = todo.IsCompleted;
33                 _db.SaveChanges();
34             }
35             return td;
36         }
37
38         public void Delete(int id)
39         {
40             _db.Todos.Remove(new Todo {Id = id});
41             _db.SaveChanges();
42         }
43     }
```

```
@Service
public class TodosService {

    private final TodosRepository _db;

    public TodosService(TodosRepository repository) { _db = repository; }

    public List<Todo> GetAll(){ return _db.findAll(); }
    public Optional<Todo> Get(int id){
        return _db.findById(id); }

    public Todo create(Todo todo) {
        todo = _db.saveAndFlush(todo);
        return todo;
    }

    public Todo update(int id, Todo todo) {
        Optional<Todo> td = _db.findById(id);
        if (td.isPresent()){
            td.get().setCompleted(todo.isCompleted());
            td.get().setTitle(todo.getTitle());
            _db.saveAndFlush(td.get());
        }
        return td.get();
    }

    public void delete(int id){
        _db.deleteById(id);
    }
}
```

CALL EXTERNAL API: C# | JAVA

```
public class ShowsController : Controller
{
    dragosh, 4 hours ago • webapi wip
    [HttpGet]
    0 references
    public async Task<ActionResult> Get()
    {
        var client = new HttpClient();
        var result = await client.GetAsync("http://jsnoise.herokuapp.com/api/showslist");
        return Ok(await result.Content.ReadAsStringAsync());
    }
}
```

```
package com.todos.api.controllers;

import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.client.RestTemplate;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class ShowsController {

    @GetMapping(value = "api/shows")
    public ResponseEntity<?> get(){
        RestTemplate restTemplate = new RestTemplate();
        return ResponseEntity.ok(
            restTemplate
                .getForEntity( url: "http://jsnoise.herokuapp.com/api/showslist", String.class)
                .getBody());
    }
}
```

SCHEDULED JOBS: C# | JAVA

```
0 references
33     public void ConfigureServices(IServiceCollection services)
34     {
35         services.AddHostedService<SimpleJob>();
36     }
```

```
1 reference
11     public class SimpleJob : IHostedService, IDisposable
12     {
13         Timer _timer;
14
15         public Task StartAsync(CancellationToken cancellationToken)
16         {
17             _timer = new Timer(DoWork, null, TimeSpan.Zero,
18                               TimeSpan.FromSeconds(5));
19
20             return Task.CompletedTask;
21         }
22
23         private void DoWork(object state)
24         {
25             Console.WriteLine("Job executed at: " + DateTime.Now);
26         }
27
28         public Task StopAsync(CancellationToken cancellationToken)
29         {
30             _timer?.Change(Timeout.Infinite, 0);
31             return Task.CompletedTask;
32         }
33     }
```

```
@SpringBootApplication
@EnableScheduling
public class ApiApplication {
```

```
Run | Debug
public static void main(final String[] args) {
```

```
@Component
public class SimpleJob {
    //void return type && no parameters
    //@Scheduled(cron = "0 15 10 15 * ?")
    @Scheduled( fixedRate = 1000, initialDelay = 100)
    public void Execute(){
        System.out.println("Job executed at: " + new Date());
    }
}
```

TESTING CONTROLLERS C#

Using xUnit and Moq (for mocking :D)

```
[Fact]
public void Todos_CanGetAllTodos()
{
    // Arrange
    var todoServiceMock = new Mock<TodosService>();
    var todo = new Todo{Id=1,Title = "Mocked todo"};

    todoServiceMock.Setup( expression: s=> s.GetAll())
        .Returns(new List<Todo>{todo})
        .Verifiable();
    var ctrl = new TodosController(todoServiceMock.Object);

    // Act
    var result = ctrl.Get() as ActionResult<List<Todo>>;
    var data = result.Value as IEnumerable<Todo>;

    // Assert
    Assert.NotNull(data);
    Assert.True( condition: data.Any());
    Assert.Equal( expected: data.First().Title , actual: todo.Title);
    todoServiceMock.Verify();
}
```

TESTING CONTROLLERS JAVA

Using jUnit5 and Mockito (for mocking :D)

```
@RunWith(SpringRunner.class)
@WebMvcTest(TodosController.class)
//@TestPropertySource(
//    locations = "classpath:application-integrationtest.properties")
public class TodosControllerTest {
    @Autowired private MockMvc mvc;
    @MockBean private TodosService service;

    @Test
    public void ca_call_get() throws Exception {
        Todo todo = Todo.builder().Id(1).title("first todo").build();
        List<Todo> todos = Arrays.asList(todo);
        given(service.getAll()).willReturn(todos);
        mvc.perform(get(urlTemplate: "/api/todos")
                    .contentType(MediaType.APPLICATION_JSON))
                    .andExpect(status().isOk())
                    .andExpect(MockMvcResultMatchers.jsonPath(expression: "$", hasSize(1)))
                    .andExpect(MockMvcResultMatchers.jsonPath(expression: "$[0].title", Is.is(todo.getTitle())));
    }
}
```

MEMORY FOOTPRINT: C#

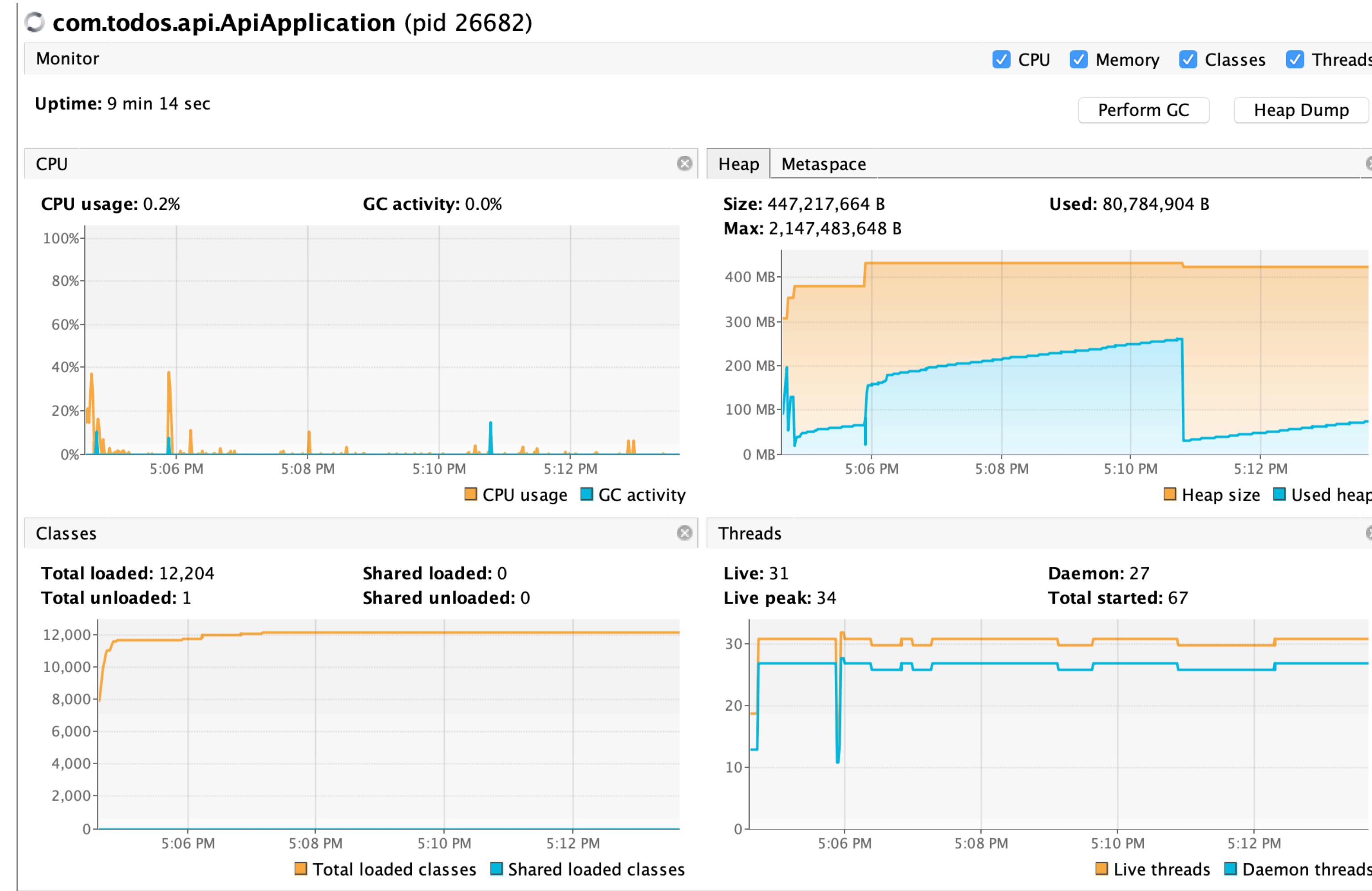
My process used working set 82.016 MB of working set

My process used working set 83.532 MB of working set

My process used working set 65.396 MB of working set

My process used working set 53.072 MB of working set

MEMORY FOOTPRINT: JAVA



SWAGGER C# WEBAPI

dotnet add package Swashbuckle.AspNetCore --version 5.0.0-rc4

//Inside the startup class add:

using Microsoft.OpenApi.Models;

Inside the ConfigureServices method:

```
// Register the Swagger generator, defining 1 or more Swagger documents
services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "Todos API", Version = "v1" });
});
```

Inside the Configure method:

```
/ Enable middleware to serve generated Swagger as a JSON endpoint.
app.UseSwagger();

// Enable middleware to serve swagger-ui (HTML, JS, CSS, etc.),
// specifying the Swagger JSON endpoint.
app.UseSwaggerUI(c =>
{
    c.SwaggerEndpoint("/swagger/v1/swagger.json", "Todos API V1");
    //next line to use the swagger as root page
    c.RoutePrefix = string.Empty;

});
```

SWAGGER C# WEBAPI

The screenshot shows the Swagger UI interface for a "Todos API V1". The top navigation bar includes a back/forward button, a refresh button, and a URL field showing "localhost:5000/index.html". To the right of the URL are various browser extension icons. The main header features the "Swagger" logo and the text "Supported by SMARTBEAR". A dropdown menu labeled "Select a definition" is set to "Todos API V1".

The main content area displays the "Todos API" documentation. It includes the version "v1 OAS3" and the URL "/swagger/v1/swagger.json".

The "Shows" section contains one endpoint:

```
GET /api/Shows
```

The "Todos" section contains five endpoints, each with a colored button indicating the HTTP method:

- GET /api/Todos
- POST /api/Todos
- GET /api/Todos/{id}
- PUT /api/Todos/{id}
- DELETE /api/Todos/{id}

The "Schemas" section lists one schema:

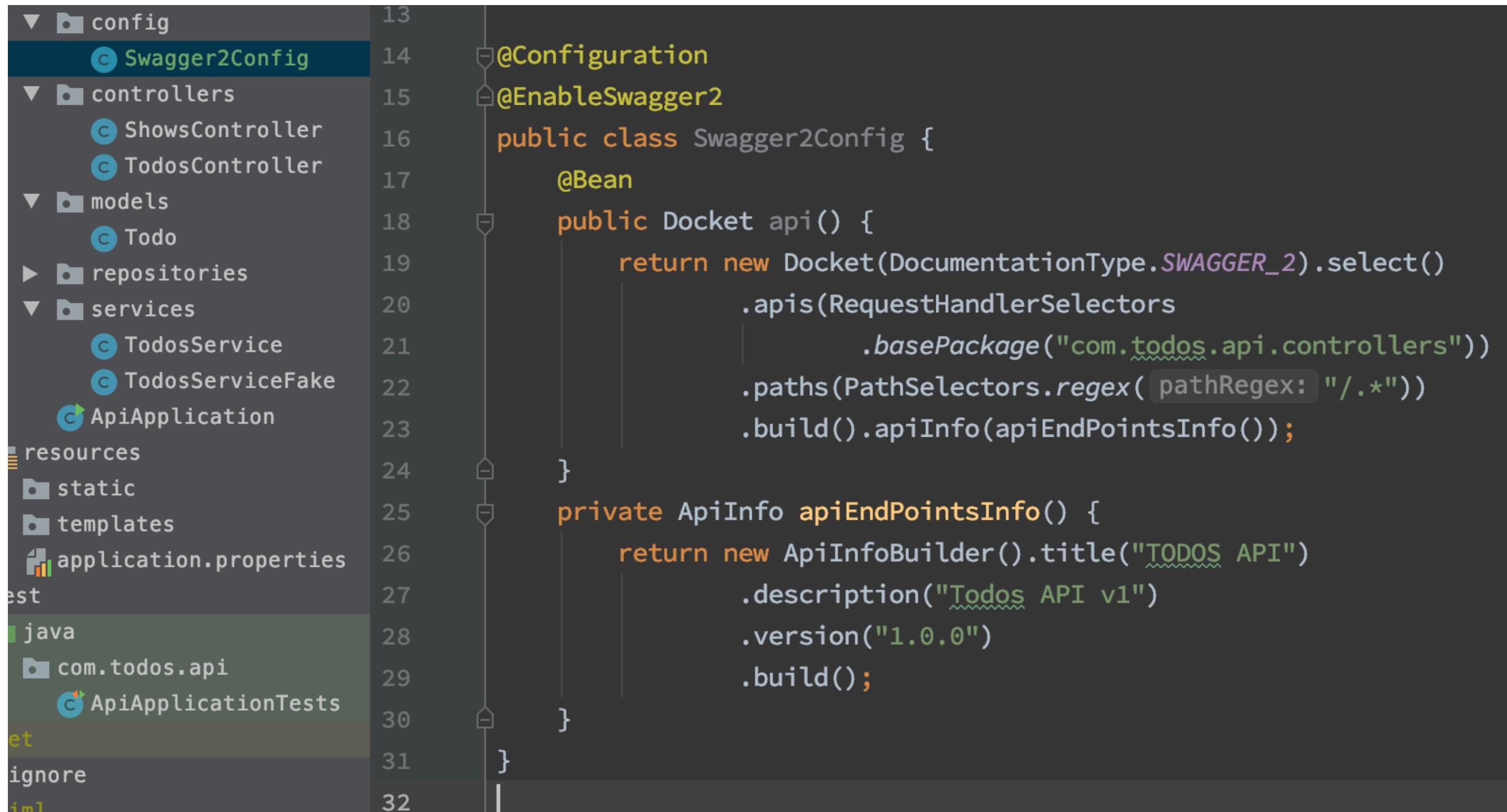
```
Todo >
```

SWAGGER JAVA SPRING BOOT

Inside pom.xml add the Springfox dependencies

```
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.8.0</version>
</dependency>

<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.8.0</version>
</dependency>
</dependencies>
```



The screenshot shows a Java code editor with the following code:

```
13  @Configuration
14  @EnableSwagger2
15  public class Swagger2Config {
16      @Bean
17      public Docket api() {
18          return new Docket(DocumentationType.SWAGGER_2).select()
19              .apis(RequestHandlerSelectors
20                  .basePackage("com.todos.api.controllers"))
21              .paths(PathSelectors.regex( pathRegex: "/.*"))
22              .build().apiInfo(apiEndPointsInfo());
23      }
24
25      private ApiInfo apiEndPointsInfo() {
26          return new ApiInfoBuilder().title("TODOS API")
27              .description("Todos API v1")
28              .version("1.0.0")
29              .build();
30      }
31  }
32
```

The code defines a `Swagger2Config` class annotated with `@Configuration` and `@EnableSwagger2`. It contains a `@Bean` method that returns a `Docket` object. The `Docket` is configured to select APIs from the `com.todos.api.controllers` package and paths matching `/.*`. It then builds an `ApiInfo` object with the title "TODOS API", description "Todos API v1", and version "1.0.0".

SWAGGER JAVA SPRING BOOT

.....

The screenshot shows the Swagger UI interface for a Todos API. The top navigation bar includes a back arrow, forward arrow, refresh button, and a search bar with placeholder text "localhost:8080/swagger-ui.html#/todos-con...". The title bar says "swagger" and "Select a spec default". Below the header, the main title is "TODOS API 1.0.0" with a note "[Base URL: localhost:8080/]" and a link "<http://localhost:8080/v2/api-docs>". A sub-header "Todos API v1" is present. The interface is organized into sections: "shows-controller Shows Controller" containing a single GET method for "/api/shows"; and "todos-controller Todos Controller" containing five methods: GET for "/api/todos", POST for "/api/todos", GET for "/api/todos/{id}", PUT for "/api/todos/{id}", and DELETE for "/api/todos/{id}". Each method is represented by a colored button (blue for GET, green for POST, blue for GET with id, orange for PUT, red for DELETE) followed by its path and verb. At the bottom, there is a "Models" section with a right-pointing arrow.

TODOS API 1.0.0
[Base URL: localhost:8080/]
<http://localhost:8080/v2/api-docs>

Todos API v1

shows-controller Shows Controller

GET /api/shows get

todos-controller Todos Controller

GET /api/todos get

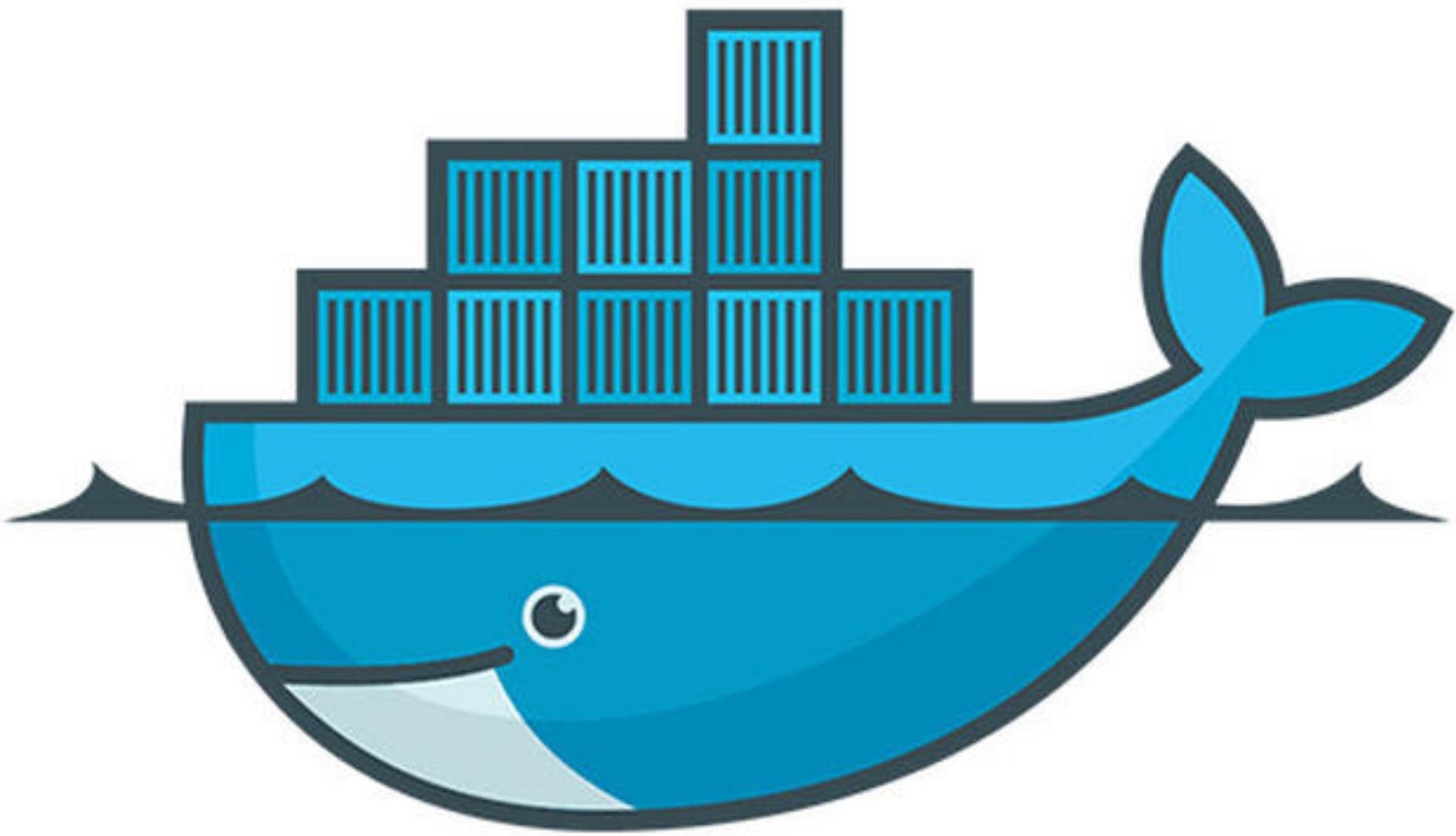
POST /api/todos post

GET /api/todos/{id} get

PUT /api/todos/{id} put

DELETE /api/todos/{id} delete

Models >



docker

DOCKERFILE: C# | JAVA

docker build -t webapi .

docker run -p 5000:80 webapi

docker build -t spring .

docker run -p 8080:8080 spring

```
1
2 FROM mcr.microsoft.com/dotnet/core/sdk:2.2 AS build-env
3 WORKDIR /app
4
5 # Copy csproj and restore as distinct layers
6 COPY *.csproj ./
7 RUN dotnet restore
8
9 # Copy everything else and build
10 COPY . ./
11 RUN dotnet publish -c Release -o out
12
13 # Build runtime image
14 FROM mcr.microsoft.com/dotnet/core/aspnet:2.2
15 WORKDIR /app
16 COPY --from=build-env /app/out .
17 ENTRYPOINT ["dotnet", "api.dll"]
```

```
1
2 FROM maven:3.5.2-jdk-8-alpine AS MAVEN_BUILD
3
4 COPY pom.xml /build/
5 COPY src /build/src/
6 WORKDIR /build/
7 RUN mvn package -P prod
8
9 FROM openjdk:8-jre-alpine
10 WORKDIR /app
11 COPY --from=MAVEN_BUILD /build/target/api-0.0.1-SNAPSHOT.jar /app/
12 ENTRYPOINT ["java","-Dspring.profiles.active=prod ", "-jar", "api-0.0.1-SNAPSHOT.jar"]
```

DOCKER-COMPOSE: C# | JAVA

docker-compose up

```
1  version: "3"
2  services:
3    db:
4      image: "postgres"
5      # ports:
6      #       - "5432:5432"
7      environment:
8        POSTGRES_USER: postgres
9        POSTGRES_PASSWORD: postgres
10     volumes:
11       - ./postgres-data:/var/lib/postgresql/data
12     networks:
13       - cs-network
14
15   web:
16     build: .
17     ports:
18       - "5000:80"
19     depends_on:
20       - db
21     networks:
22       - cs-network
23     environment:
24       DATABASE_URL: Host=db;Port=5432;Username=postgres;Password=password;Database=todos_cs;
25   networks:
26     cs-network:
27       driver: bridge
```

docker-compose up

```
1  version: "3"
2  services:
3    db:
4      image: "postgres"
5      environment:
6        POSTGRES_DB: postgres
7        POSTGRES_USER: postgres
8        POSTGRES_PASSWORD: postgres
9
10   volumes:
11     - ./postgres-data:/var/lib/postgresql/data
12     #- ./init.sql:/docker-entrypoint-initdb.d/10-db.sql
13
14   networks:
15     - java-network
16
17
18   web:
19     #build: .
20     image: spring
21     ports:
22       - "8080:8080"
23     depends_on:
24       - db
25     environment:
26       - SPRING_DATASOURCE_URL= jdbc:postgresql://db:5432/postgres
27       - SPRING_DATASOURCE_USERNAME=postgres
28       - SPRING_DATASOURCE_PASSWORD=postgres
29     networks:
30       - java-network
31
32   networks:
33     java-network:
34       driver: bridge
```

HEROKU

 **HEROKU** Products ▾ Marketplace ▾ Pricing Documentation Support More ▾ SEARCH



DEVELOPERS

Focus on your apps

Invest in apps, not ops. Heroku handles the hard stuff — patching and upgrading, 24/7 ops and security, build systems, failovers, and more — so your developers can stay focused on building great apps.

[SIGN UP FOR FREE](#)

[Explore the Heroku Platform](#)



HEROKU WEBAPI

<https://github.com/imhotepper/HerokuPgSqlConnectionParser>



HerokuPGParser 1.0.2

Package Description

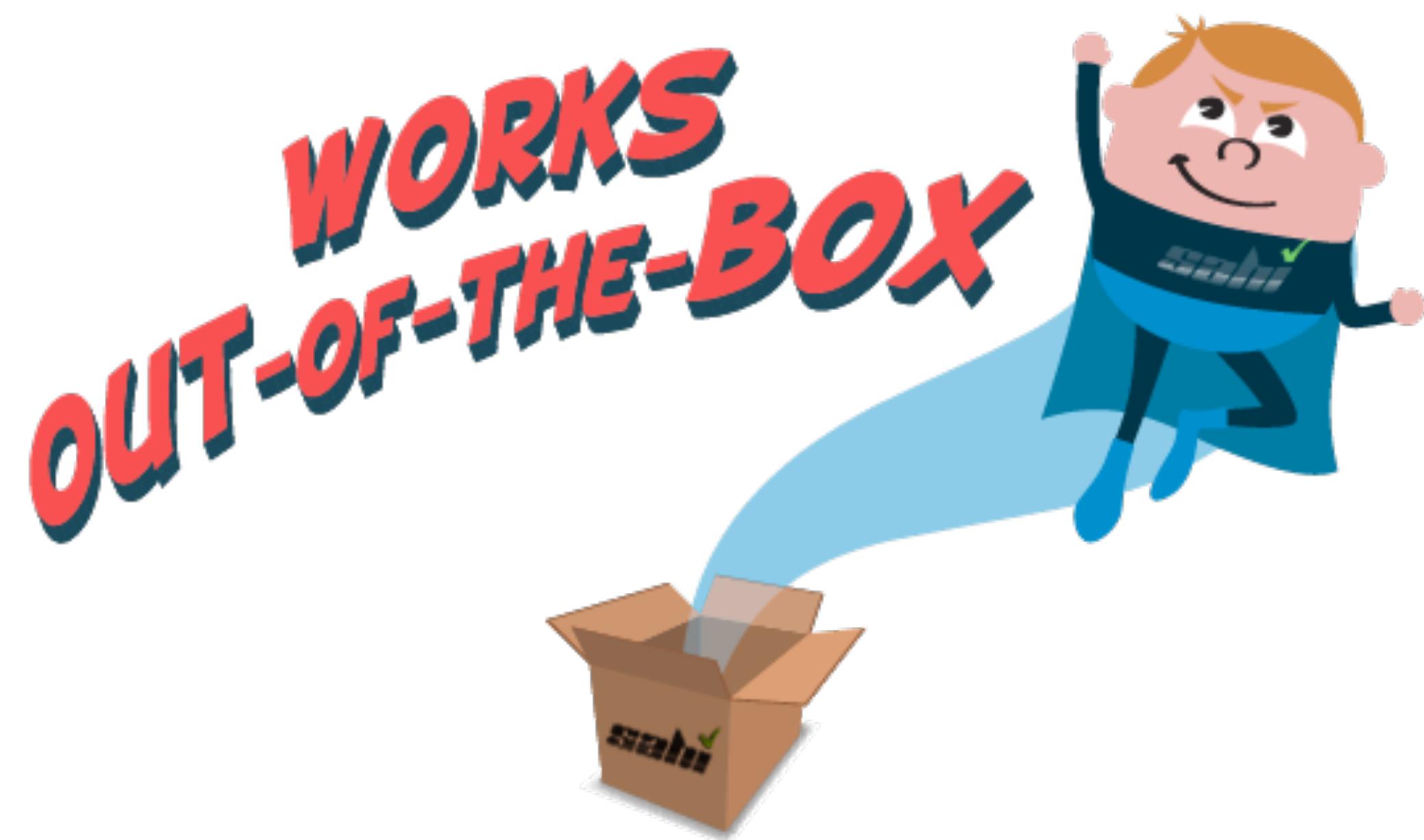
Package Manager .NET CLI PackageReference Paket CLI

> dotnet add package HerokuPGParser --version 1.0.2



```
C# Startup.cs X
api > C# Startup.cs > {} api > 🏛 api.Startup > ⚙ ConfigureServices(IServiceCollection services)
29   |
30   |     2 references
31   |     public IConfiguration Configuration { get; }
32   |     // This method gets called by the runtime. Use this method to add services to the container.
33   |     0 references
34   |     public void ConfigureServices(IServiceCollection services)
35   |     {
36   |         var conStr = Configuration.GetConnectionString("DefaultConnection");
37   |         var pgConn = Environment.GetEnvironmentVariable("DATABASE_URL");
38   |
39   |         if (!string.IsNullOrWhiteSpace(pgConn))
        |             conStr = HerokuPGParser.ConnectionHelper.BuildExpectedConnectionString(pgConn);
```

HEROKU SPRING-BOOT



<https://ct-todos-cs.herokuapp.com/>

Deployment method

Heroku Git Use Heroku CLI GitHub Connect to GitHub Container Registry Use Heroku CLI

Connect to GitHub

Search for a repository to connect to

Connect this app to GitHub to enable code diffs and deploys.

imhotepper ct-todos-cs Search

Missing a GitHub organization? [Ensure Heroku Dashboard has team access.](#)

imhotepper/ct-todos-cs Connect

Add-ons

The add-on heroku-postgresql has been installed. Check out the documentation in its [Dev Center article](#) to get started.

Quickly add add-ons from Elements

Heroku Postgres Attached as DATABASE Hobby Dev Free \$0.00

Estimated Monthly Cost

Buildpacks

Buildpacks are scripts that are run when your app is deployed. They are used to install dependencies for your app and configure your environment. [Find new buildpacks on Heroku Elements](#)

Add buildpack

https://github.com/jincod/dotnetcore-buildpack X

<https://github.com/jincod/dotnetcore-buildpack>

Buildpacks

Buildpacks are scripts that are run when your app is deployed. They are used to install dependencies for your app and configure your environment. [Find new buildpacks on Heroku Elements](#)

Add buildpack

https://github.com/jincod/dotnetcore-buildpack X

<https://ct-todos-java.herokuapp.com/swagger-ui.html>

.....

Deployment method

 Heroku Git
Use Heroku CLI

 GitHub Connected 

 Container Registry
Use Heroku CLI

App connected to GitHub

Code diffs, manual and auto deploys are available for this app.

Connected to  imhotepper/ct-todos-java by  imhotepper

[Disconnect...](#)

-  Releases in the [activity feed](#) link to GitHub to view commit diffs
-  Automatically deploys from  master

Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

 Automatic deploys from  master are enabled

Every push to  master will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch in GitHub is always in a deployable state and any tests have passed before you push. [Learn more](#).

Wait for CI to pass before deploy
Only enable this option if you have a Continuous Integration service configured on your repo.

[Disable Automatic Deploys](#)

[Find more add-ons](#)

The add-on  heroku-postgresql has been installed. Check out the documentation in its [Dev Center](#) article to get started.

 Quickly add add-ons from Elements

 Heroku Postgres 

Attached as DATABASE 

Hobby Dev Free 

Estimated Monthly Cost  \$0.00

SIZE & MEMORY FOOTPRINT

.....

```
----> Discovering process types  
      Procfile declares types -> web  
----> Compressing...  
      Done: 68.5M  
----> Launching...  
      Released v7  
      https://ct-todos-cs.herokuapp.com/ deployed to Heroku
```

```
----> Discovering process types  
      Procfile declares types -> (none)  
      Default types for buildpack -> web  
----> Compressing...  
      Done: 90M  
----> Launching...  
      Released v5  
      https://ct-todos-java.herokuapp.com/ deployed to Heroku
```

```
heroku labs:enable log-runtime-metrics -a [appname]
```

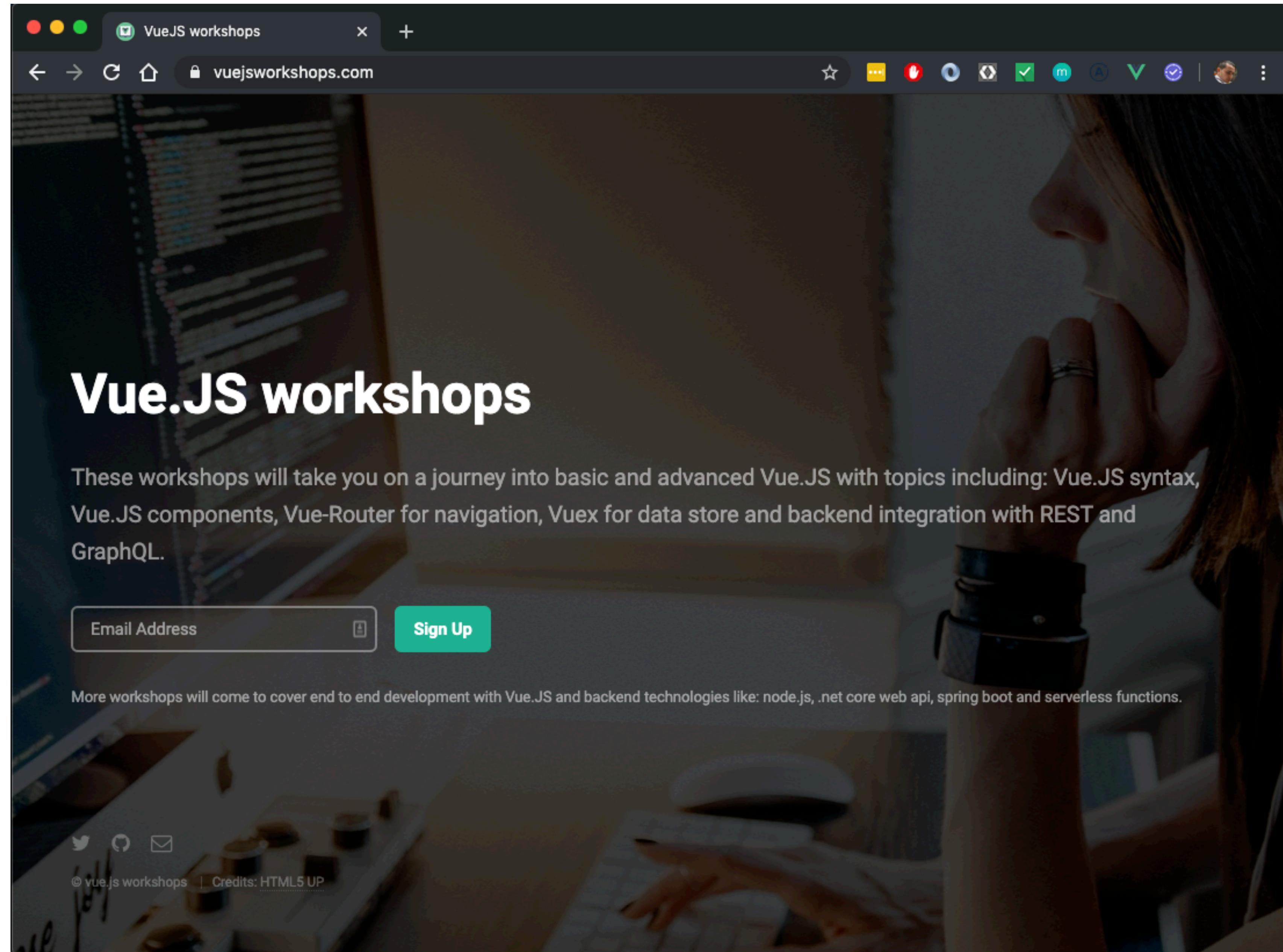
```
2019-11-25T15:09:25.858722+00:00 heroku[web.1]: source=web.1 dyno=heroku.153885564.daf60bcd-41d2-4e4c-830c-efea0060e81e sample#load_avg_1m=0.01  
sample#load_avg_5m=0.03  
2019-11-25T15:09:25.858722+00:00 heroku[web.1]: source=web.1 dyno=heroku.153885564.daf60bcd-41d2-4e4c-830c-efea0060e81e sample#memory_total=56.16MB  
sample#memory_rss=50.55MB sample#memory_cache=5.61MB sample#memory_swap=0.00MB sample#memory_pgpgin=173325pages sample#memory_pgpgout=158948pages  
sample#memory_quota=512.00MB
```

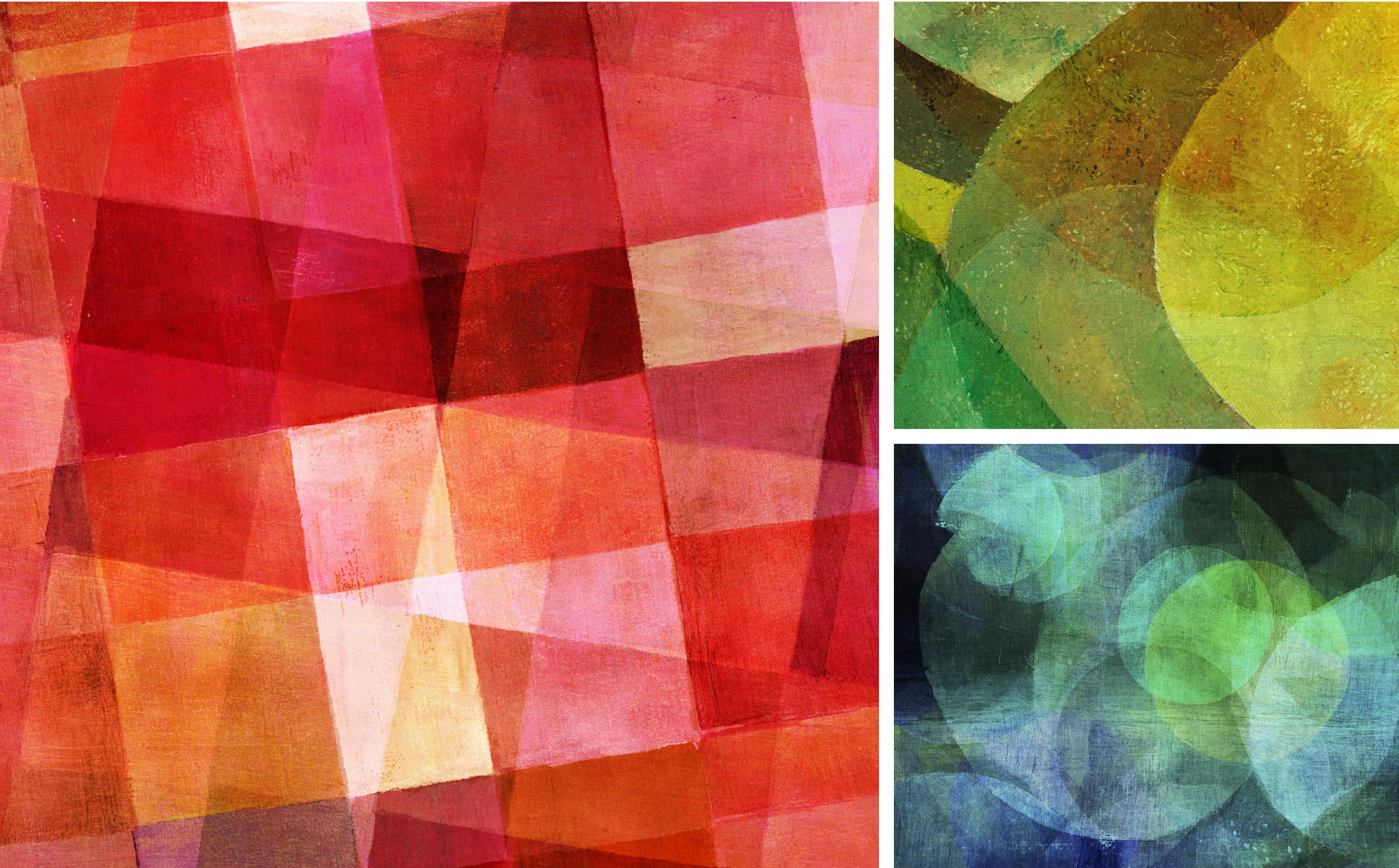
```
2019-11-25T15:07:45.303183+00:00 heroku[web.1]: source=web.1 dyno=heroku.153886854.22ddfbf8-4f5f-4596-b672-b185e3e411b1 sample#load_avg_1m=0.00  
2019-11-25T15:07:45.303284+00:00 heroku[web.1]: source=web.1 dyno=heroku.153886854.22ddfbf8-4f5f-4596-b672-b185e3e411b1 sample#memory_total=301.38MB  
sample#memory_rss=301.35MB sample#memory_cache=0.03MB sample#memory_swap=0.00MB sample#memory_pgpgin=88632pages sample#memory_pgpgout=13522pages  
sample#memory_quota=512.00MB
```



RECAP

- Prerequisites
- Dependencies management
- Creating a new api
- Create CRUD Controller
- Validation
- Error handling
- Database access
- Call another API
- Scheduled Jobs
- Testing controllers
- Memory footprint
- Swagger
- Docker
- Deploy to heroku





Thank you for your time & attention!