NET CORE GENESIS

Fundamentals & Solution Generation from Database

CLI, C#.Net Core Backend, React JS UI

v1.3.2

Feb 2020

What is and Why Net Core Genesis?

When you start a new project, there are lots of common infrastructural tasks and routine developments forcing you to spend valuable time re-inventing the wheel instead of focusing on your core business.

1) Backend & UI Framework
as a ready infrastructure

Genesis helps you get jobs done ahead of your schedule by providing you ready-to-go Dot Net Core & ReactJS project solutions

2) AutoCode
as a Solution Generator

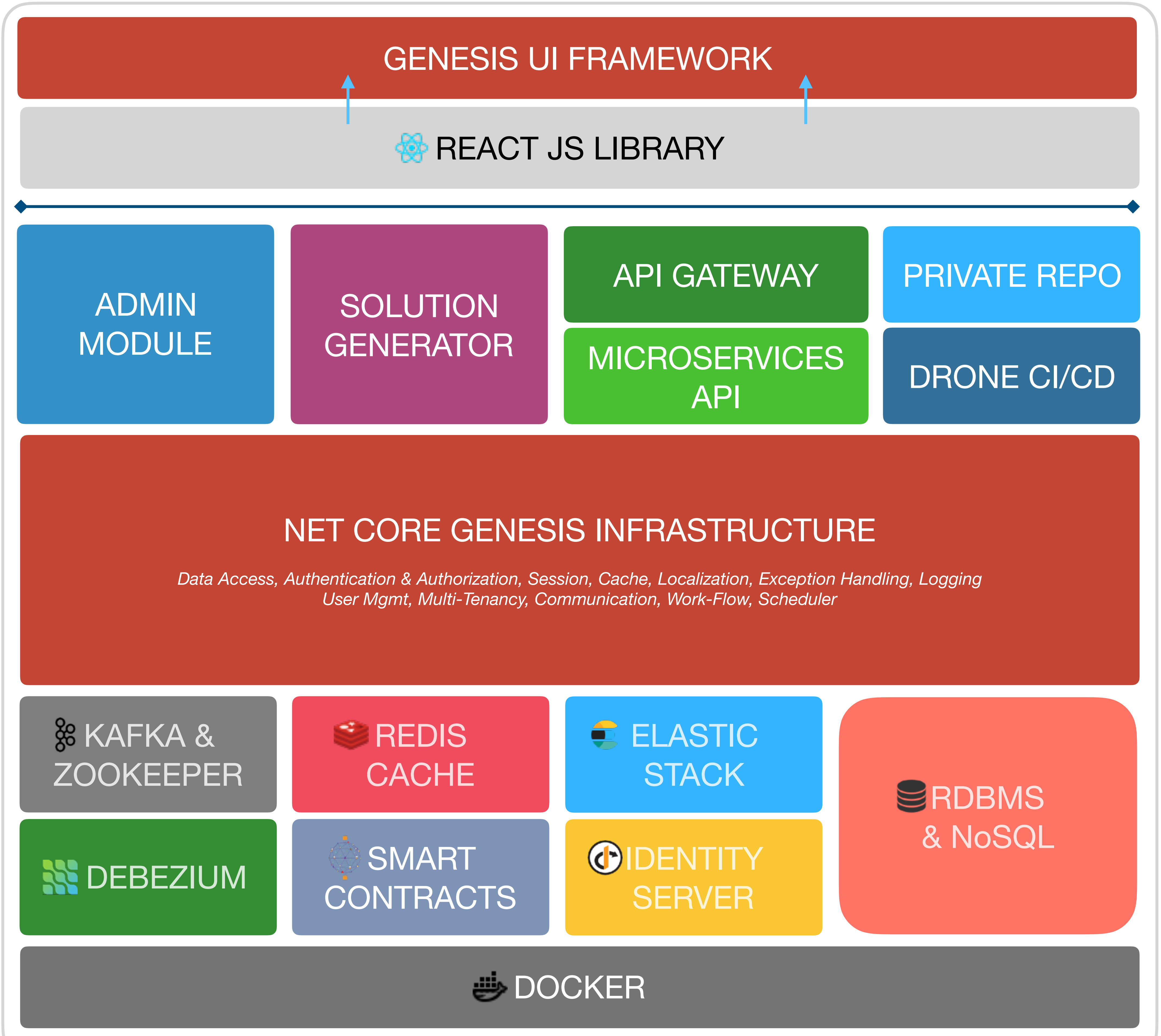
A cross platform CLI-Command Line Interface generates all the strong code necessary to bootstrap your business from Day 1.

3) DevOps Automation
as CI/CD Process

Just push your code to Git version management, Genesis platform will be handling the rest of your deployment to server

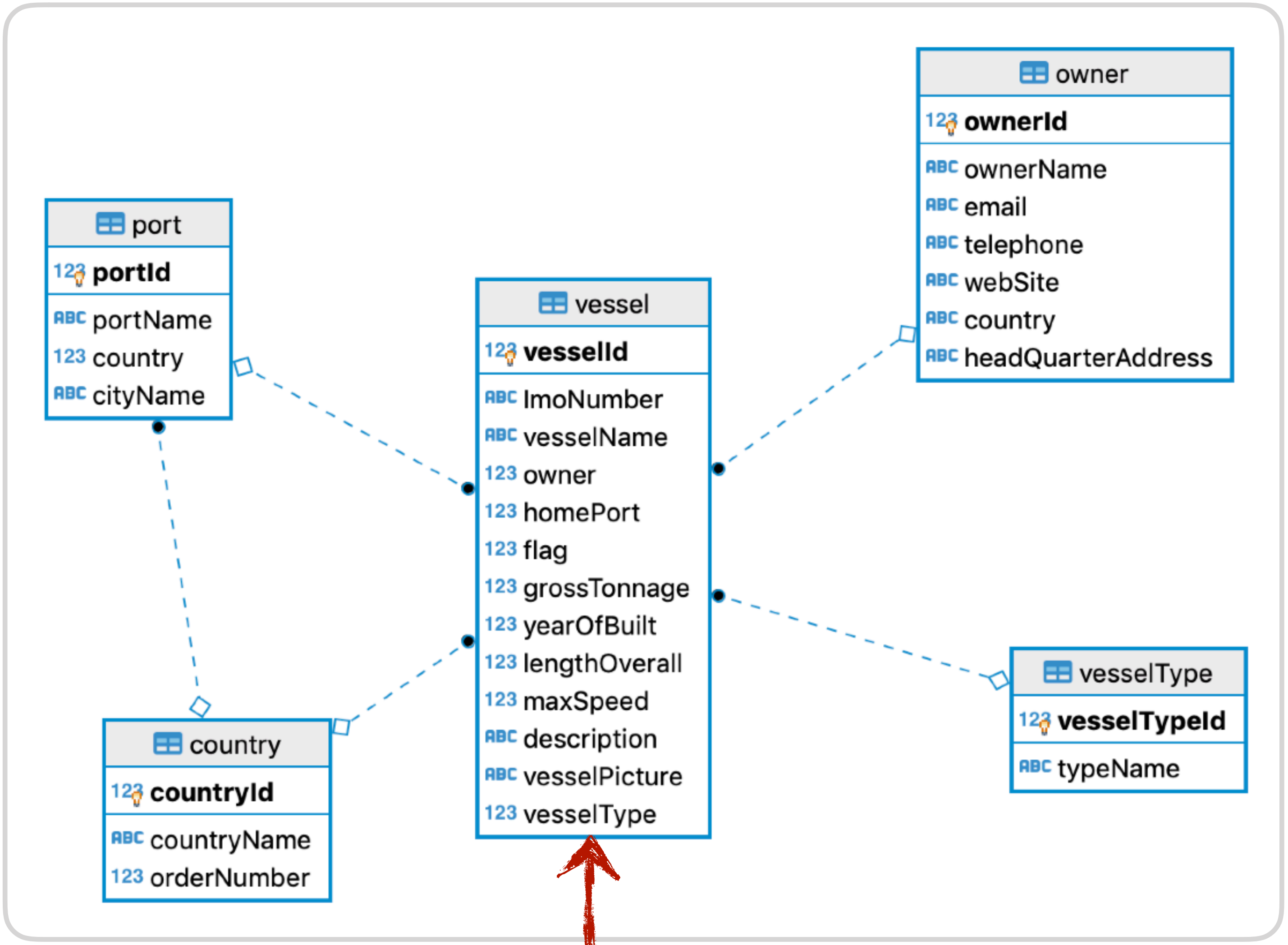
Architecture Design

NET CORE GENESIS



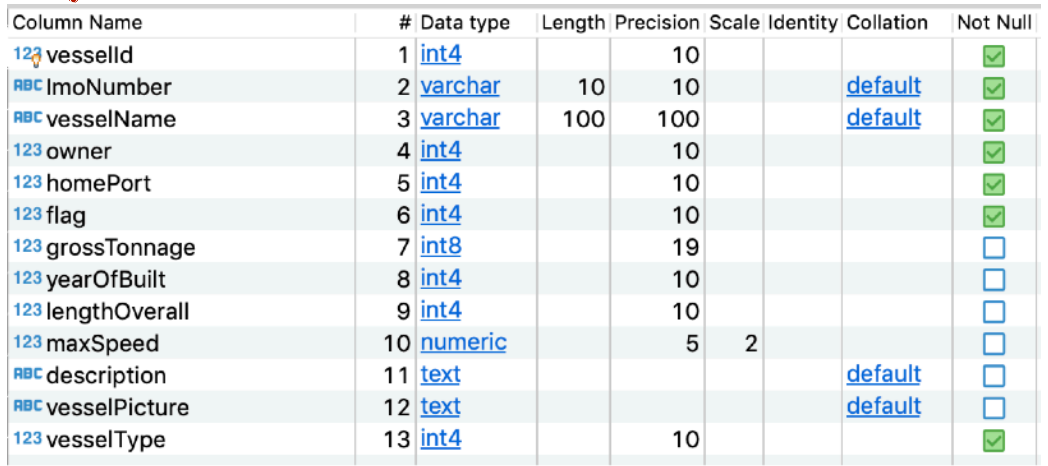
NET CORE GENESIS

Database / ER Diagram



Any business can be built on Genesis.

Just create a proper and comprehensive data model (or services) for a quick start



Column Name	#	Data type	Length	Precision	Scale	Identity	Collation	Not Null
vesselId	1	int4		10				✓
ImoNumber	2	varchar	10	10			default	✓
vesselName	3	varchar	100	100			default	✓
owner	4	int4		10				✓
homePort	5	int4		10				✓
flag	6	int4		10				✓
grossTonnage	7	int8		19				☐
yearOfBuilt	8	int4		10				☐
lengthOverall	9	int4		10				☐
maxSpeed	10	numeric		5	2			☐
description	11	text					default	☐
vesselPicture	12	text					default	☐
vesselType	13	int4		10				✓

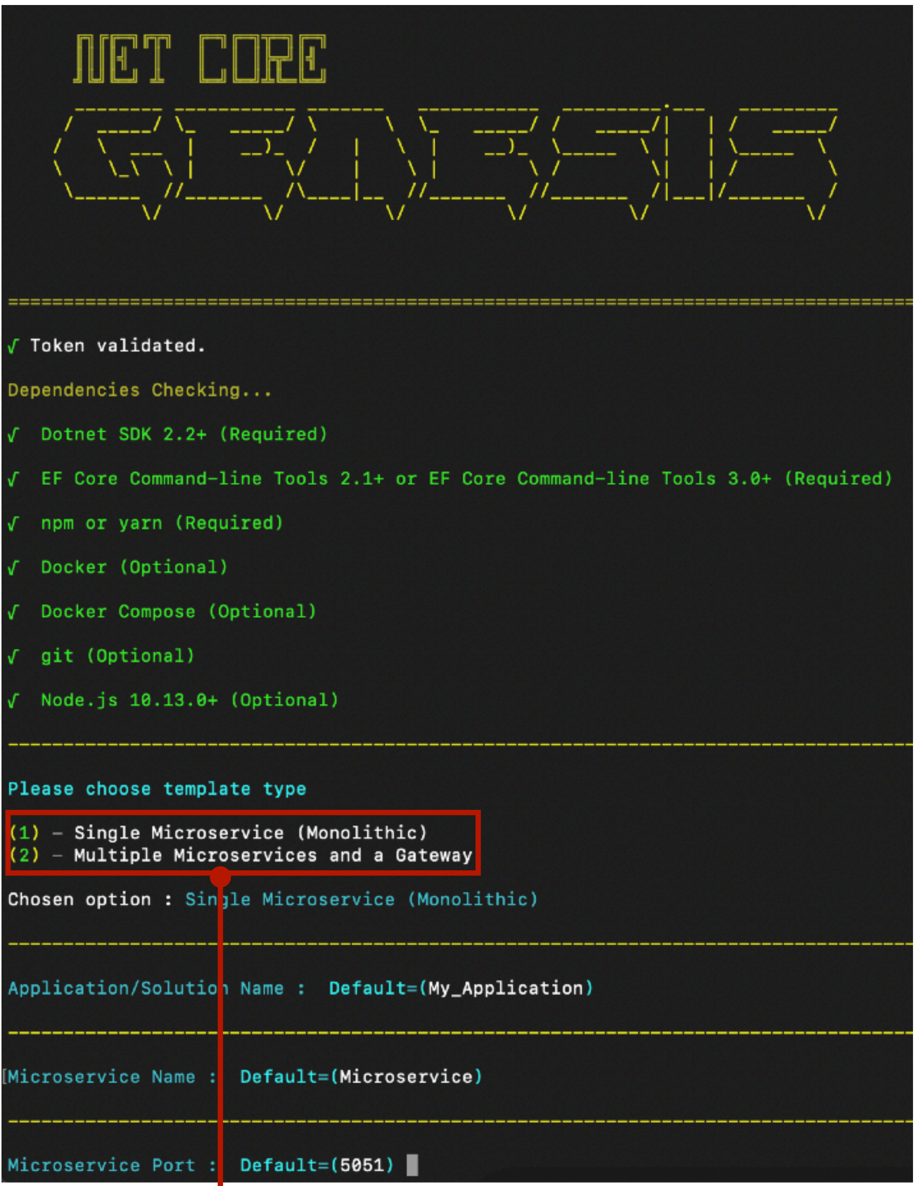
All metadata is fetched

- Table name
- Column name
- Data type
- Length
- Precision, Scale
- Primary key, Foreign key, other constraints
- Nullability and so on…

NET CORE GENESIS

Solution Generator *(CLI/Terminal based)*

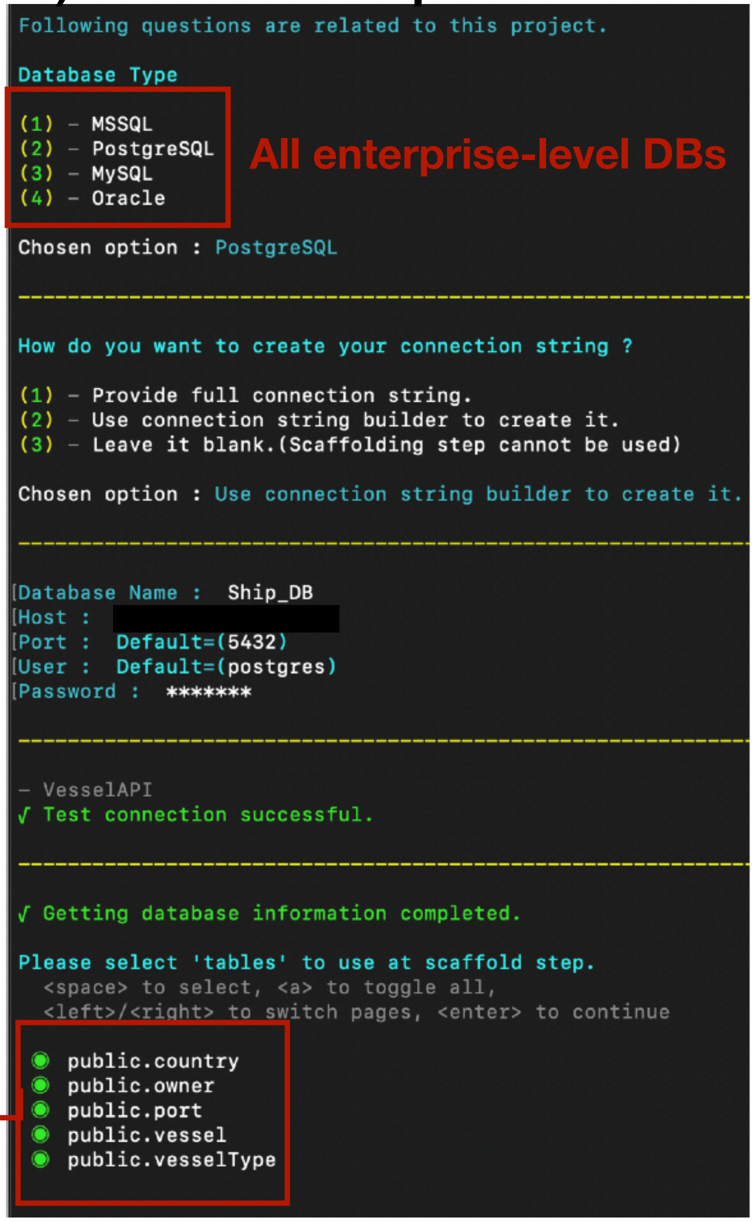
1) Start creation



```
NET CORE
GENESIS

================================================================================

√ Token validated.

Dependencies Checking...

√  Dotnet SDK 2.2+ (Required)

√  EF Core Command-line Tools 2.1+ or EF Core Command-line Tools 3.0+ (Required)

√  npm or yarn (Required)

√  Docker (Optional)

√  Docker Compose (Optional)

√  git (Optional)

√  Node.js 10.13.0+ (Optional)

--------------------------------------------------------------------------------

Please choose template type

(1) - Single Microservice (Monolithic)
(2) - Multiple Microservices and a Gateway

Chosen option : Single Microservice (Monolithic)

--------------------------------------------------------------------------------

Application/Solution Name :  Default=(My_Application)

--------------------------------------------------------------------------------

Microservice Name :  Default=(Microservice)

--------------------------------------------------------------------------------

Microservice Port :  Default=(5051)
```

Microservice architecture support

2) Provide preferences

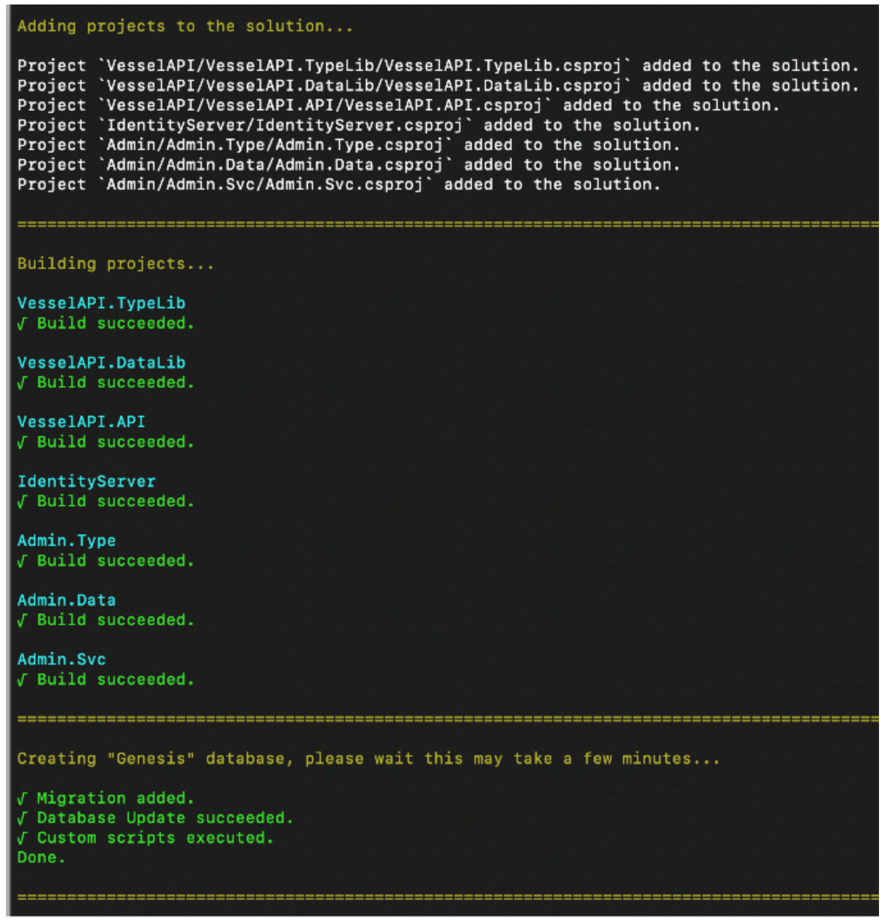


```
Following questions are related to this project.

Database Type

(1) - MSSQL
(2) - PostgreSQL
(3) - MySQL
(4) - Oracle

Chosen option : PostgreSQL

--------------------------------------------------------------------------------

How do you want to create your connection string ?

(1) - Provide full connection string.
(2) - Use connection string builder to create it.
(3) - Leave it blank.(Scaffolding step cannot be used)

Chosen option : Use connection string builder to create it.

--------------------------------------------------------------------------------

Database Name :  Ship_DB
Host :  [redacted]
Port :  Default=(5432)
User :  Default=(postgres)
Password :  *******

--------------------------------------------------------------------------------

- VesselAPI
√ Test connection successful.

--------------------------------------------------------------------------------

√ Getting database information completed.

Please select 'tables' to use at scaffold step.
  <space> to select, <a> to toggle all,
  <left>/<right> to switch pages, <enter> to continue

● public.country
● public.owner
● public.port
● public.vessel
● public.vesselType
```

All enterprise-level DBs

Choose DB tables

3) Let it be generated from end-to-end

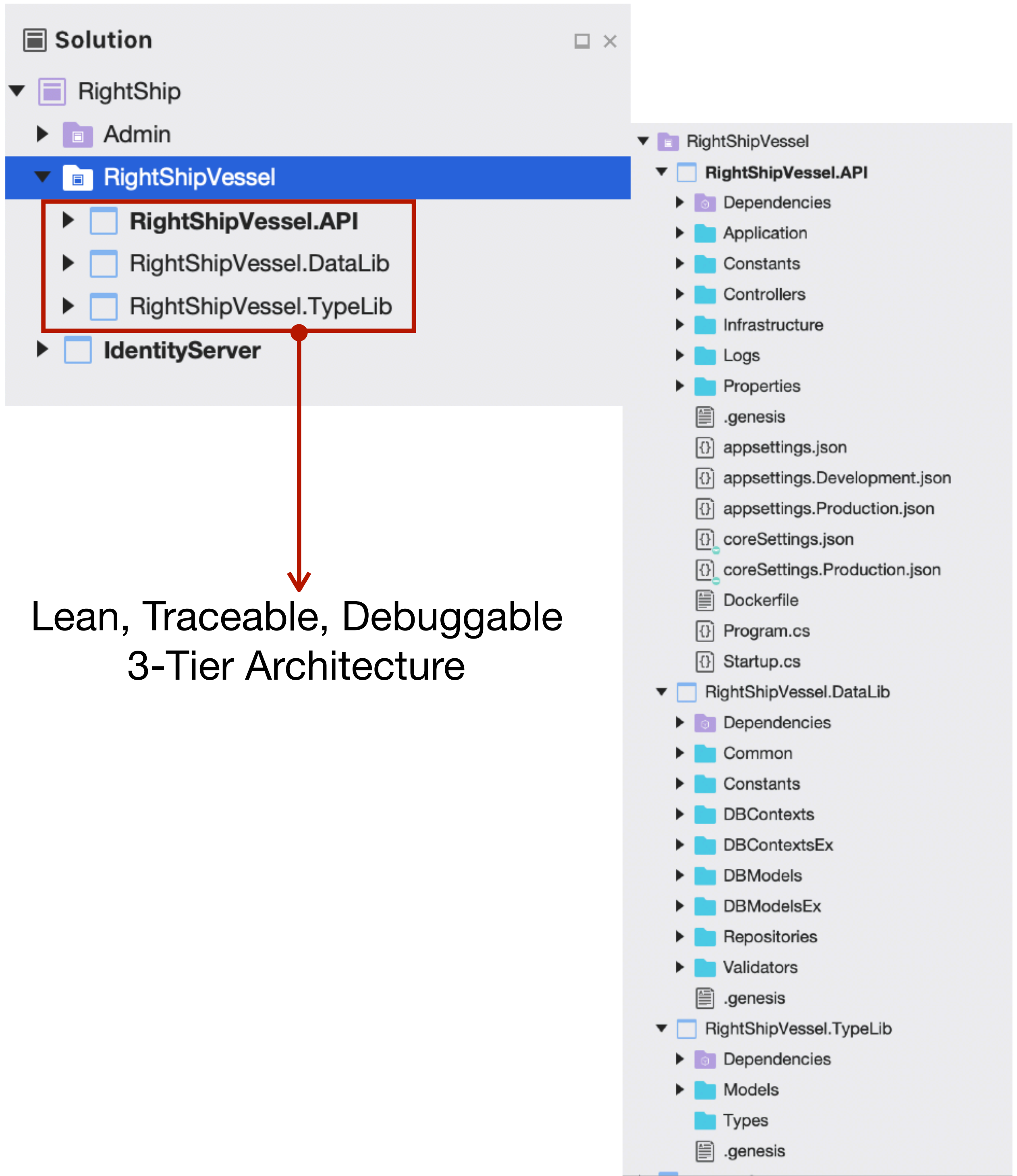


```
Adding projects to the solution...

Project `VesselAPI/VesselAPI.TypeLib/VesselAPI.TypeLib.csproj` added to the solution.
Project `VesselAPI/VesselAPI.DataLib/VesselAPI.DataLib.csproj` added to the solution.
Project `VesselAPI/VesselAPI.API/VesselAPI.API.csproj` added to the solution.
Project `IdentityServer/IdentityServer.csproj` added to the solution.
Project `Admin/Admin.Type/Admin.Type.csproj` added to the solution.
Project `Admin/Admin.Data/Admin.Data.csproj` added to the solution.
Project `Admin/Admin.Svc/Admin.Svc.csproj` added to the solution.

================================================================================

Building projects...

VesselAPI.TypeLib
√ Build succeeded.

VesselAPI.DataLib
√ Build succeeded.

VesselAPI.API
√ Build succeeded.

IdentityServer
√ Build succeeded.

Admin.Type
√ Build succeeded.

Admin.Data
√ Build succeeded.

Admin.Svc
√ Build succeeded.

================================================================================

Creating "Genesis" database, please wait this may take a few minutes...

√ Migration added.
√ Database Update succeeded.
√ Custom scripts executed.
Done.

================================================================================
```

NET CORE GENESIS

Generated Projects

1) Backend (C#.Net Core)

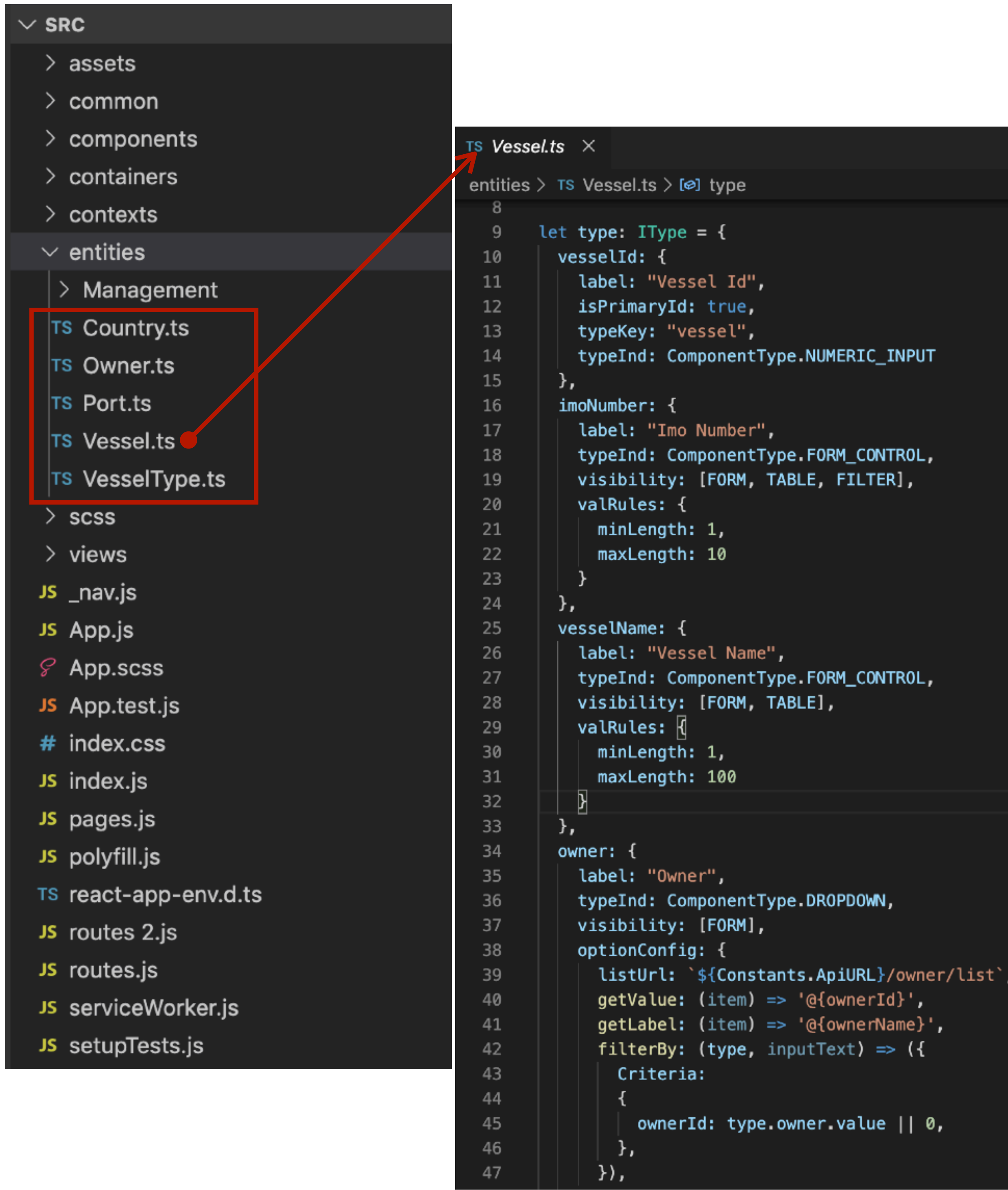


```
Solution
▼ RightShip
  ▶ Admin
  ▼ RightShipVessel
    ▶ RightShipVessel.API
    ▶ RightShipVessel.DataLib
    ▶ RightShipVessel.TypeLib
  ▶ IdentityServer
```

```
▼ RightShipVessel
  ▼ RightShipVessel.API
    ▶ Dependencies
    ▶ Application
    ▶ Constants
    ▶ Controllers
    ▶ Infrastructure
    ▶ Logs
    ▶ Properties
      .genesis
      appsettings.json
      appsettings.Development.json
      appsettings.Production.json
      coreSettings.json
      coreSettings.Production.json
      Dockerfile
      Program.cs
      Startup.cs
  ▼ RightShipVessel.DataLib
    ▶ Dependencies
    ▶ Common
    ▶ Constants
    ▶ DBContexts
    ▶ DBContextsEx
    ▶ DBModels
    ▶ DBModelsEx
    ▶ Repositories
    ▶ Validators
      .genesis
  ▼ RightShipVessel.TypeLib
    ▶ Dependencies
    ▶ Models
      Types
      .genesis
```

Lean, Traceable, Debuggable
3-Tier Architecture

2) Frontend (React JS)



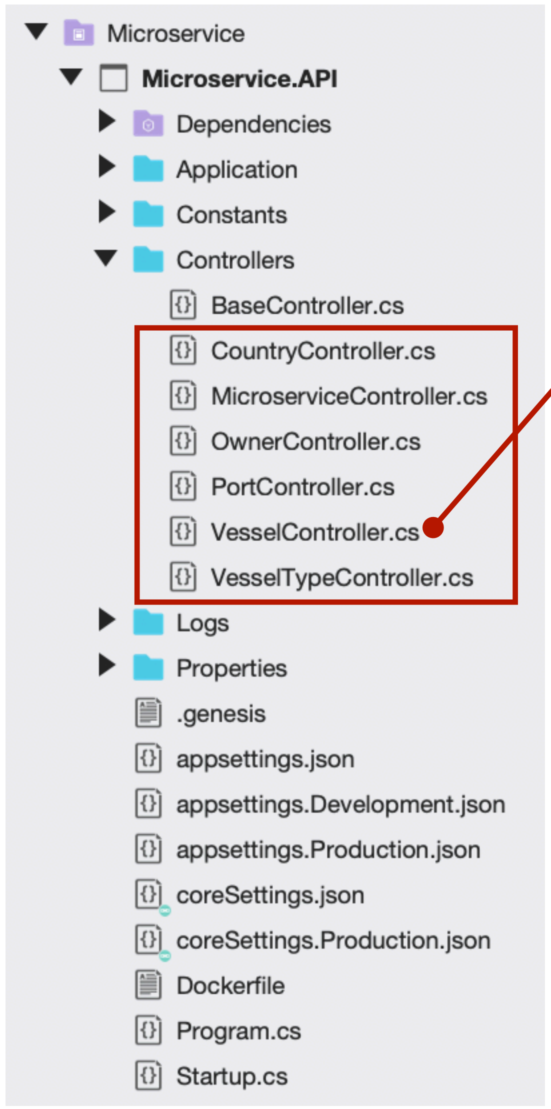
```
∨ SRC
  > assets
  > common
  > components
  > containers
  > contexts
  ∨ entities
    > Management
    TS Country.ts
    TS Owner.ts
    TS Port.ts
    TS Vessel.ts
    TS VesselType.ts
  > scss
  > views
  JS _nav.js
  JS App.js
  App.scss
  JS App.test.js
  # index.css
  JS index.js
  JS pages.js
  JS polyfill.js
  TS react-app-env.d.ts
  JS routes 2.js
  JS routes.js
  JS serviceWorker.js
  JS setupTests.js
```

```
let type: IType = {
  vesselId: {
    label: "Vessel Id",
    isPrimaryId: true,
    typeKey: "vessel",
    typeInd: ComponentType.NUMERIC_INPUT
  },
  imoNumber: {
    label: "Imo Number",
    typeInd: ComponentType.FORM_CONTROL,
    visibility: [FORM, TABLE, FILTER],
    valRules: {
      minLength: 1,
      maxLength: 10
    }
  },
  vesselName: {
    label: "Vessel Name",
    typeInd: ComponentType.FORM_CONTROL,
    visibility: [FORM, TABLE],
    valRules: {
      minLength: 1,
      maxLength: 100
    }
  },
  owner: {
    label: "Owner",
    typeInd: ComponentType.DROPDOWN,
    visibility: [FORM],
    optionConfig: {
      listUrl: `${Constants.ApiURL}/owner/list`
      getValue: (item) => '@{ownerId}',
      getLabel: (item) => '@{ownerName}',
      filterBy: (type, inputText) => ({
        Criteria:
        {
          ownerId: type.owner.value || 0,
        },
      }),
```

NET CORE GENESIS

Generated Backend / API

Controllers

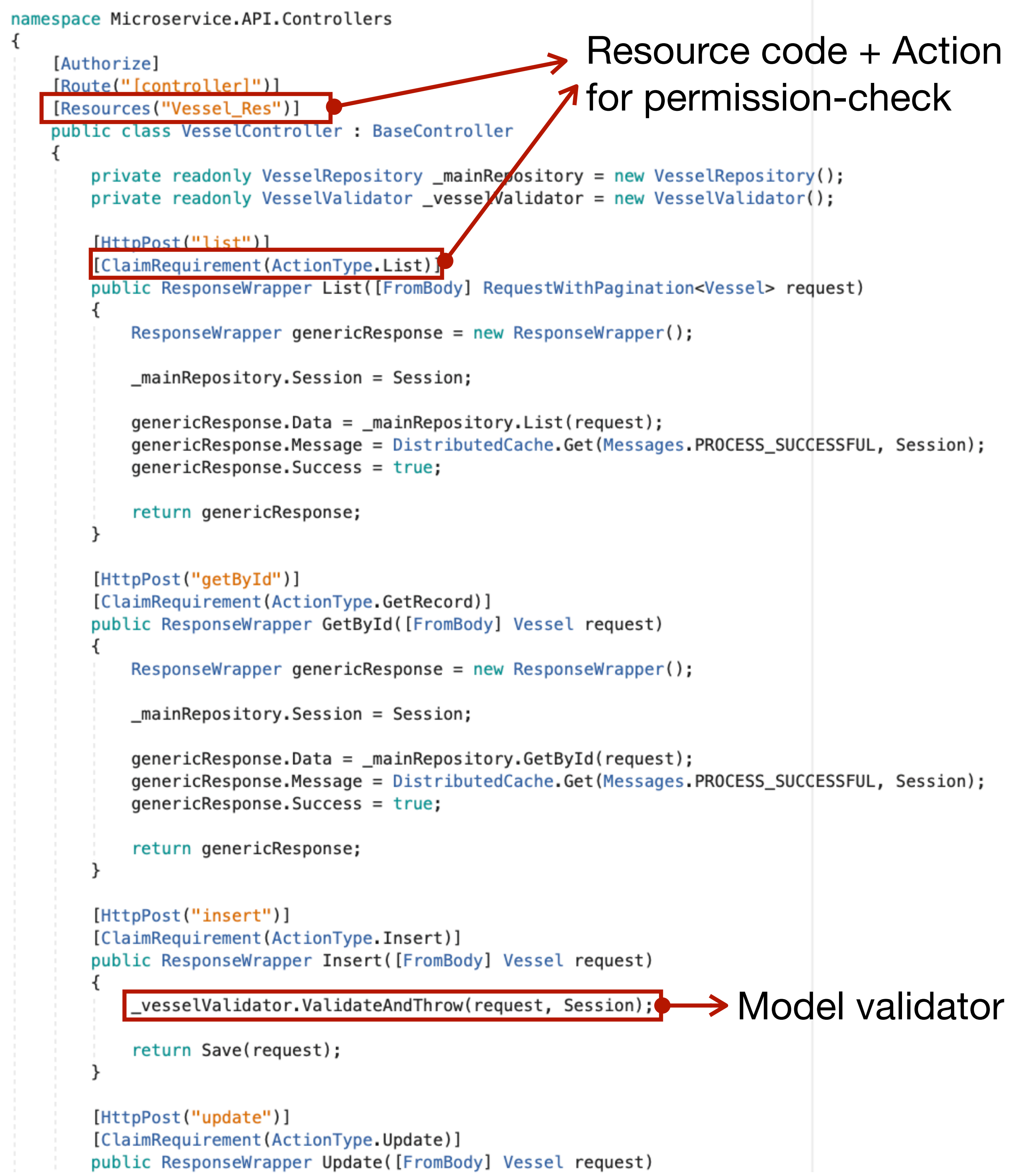


```
▼ Microservice
  ▼ Microservice.API
    ▶ Dependencies
    ▶ Application
    ▶ Constants
    ▼ Controllers
        BaseController.cs
        CountryController.cs
        MicroserviceController.cs
        OwnerController.cs
        PortController.cs
        VesselController.cs
        VesselTypeController.cs
    ▶ Logs
    ▶ Properties
      .genesis
      appsettings.json
      appsettings.Development.json
      appsettings.Production.json
      coreSettings.json
      coreSettings.Production.json
      Dockerfile
      Program.cs
      Startup.cs
```

6 methods generated for each controller

1) list
2) getById
3) insert
4) update
5) delete
6) bulkSave

Batch upload by Excel



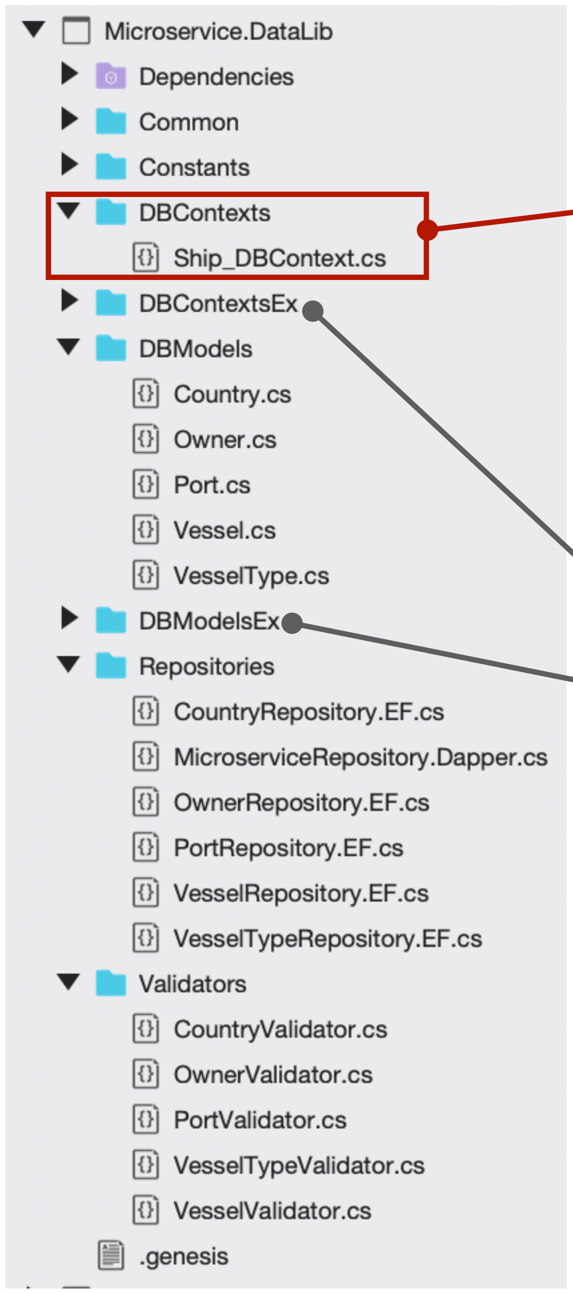
```
namespace Microservice.API.Controllers
{
    [Authorize]
    [Route("[controller]")]
    [Resources("Vessel_Res")]
    public class VesselController : BaseController
    {
        private readonly VesselRepository _mainRepository = new VesselRepository();
        private readonly VesselValidator _vesselValidator = new VesselValidator();

        [HttpPost("list")]
        [ClaimRequirement(ActionType.List)]
        public ResponseWrapper List([FromBody] RequestWithPagination<Vessel> request)
        {
            ResponseWrapper genericResponse = new ResponseWrapper();

            _mainRepository.Session = Session;

            genericResponse.Data = _mainRepository.List(request);
            genericResponse.Message = DistributedCache.Get(Messages.PROCESS_SUCCESSFUL, Session);
            genericResponse.Success = true;

            return genericResponse;
        }

        [HttpPost("getById")]
        [ClaimRequirement(ActionType.GetRecord)]
        public ResponseWrapper GetById([FromBody] Vessel request)
        {
            ResponseWrapper genericResponse = new ResponseWrapper();

            _mainRepository.Session = Session;

            genericResponse.Data = _mainRepository.GetById(request);
            genericResponse.Message = DistributedCache.Get(Messages.PROCESS_SUCCESSFUL, Session);
            genericResponse.Success = true;

            return genericResponse;
        }

        [HttpPost("insert")]
        [ClaimRequirement(ActionType.Insert)]
        public ResponseWrapper Insert([FromBody] Vessel request)
        {
            _vesselValidator.ValidateAndThrow(request, Session);

            return Save(request);
        }

        [HttpPost("update")]
        [ClaimRequirement(ActionType.Update)]
        public ResponseWrapper Update([FromBody] Vessel request)
```

Resource code + Action for permission-check

Model validator

Generated Backend / Data Layer

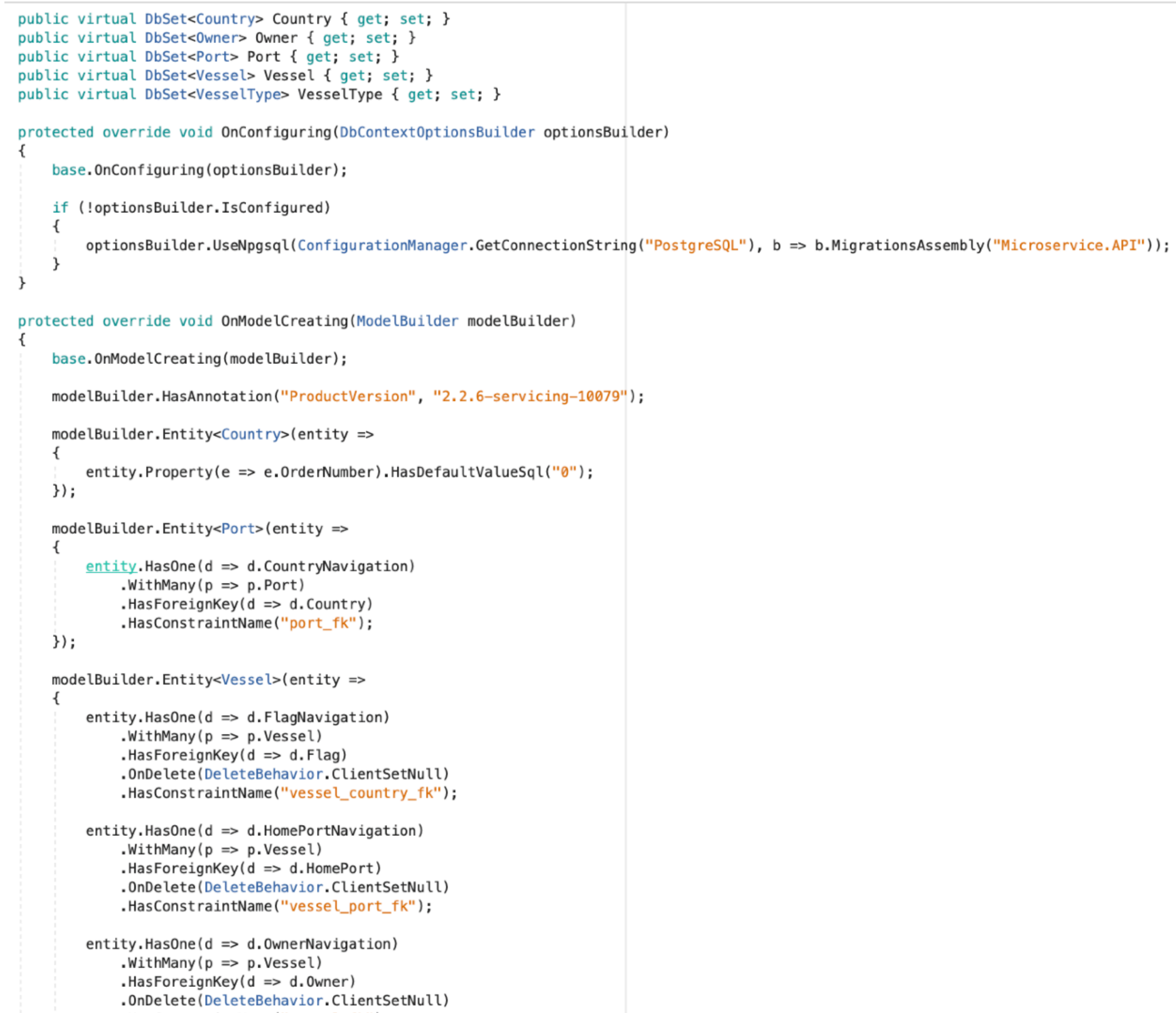
1) DBContext



- Microservice.DataLib
  - Dependencies
  - Common
  - Constants
  - DBContexts
    - Ship_DBContext.cs
  - DBContextsEx
  - DBModels
    - Country.cs
    - Owner.cs
    - Port.cs
    - Vessel.cs
    - VesselType.cs
  - DBModelsEx
  - Repositories
    - CountryRepository.EF.cs
    - MicroserviceRepository.Dapper.cs
    - OwnerRepository.EF.cs
    - PortRepository.EF.cs
    - VesselRepository.EF.cs
    - VesselTypeRepository.EF.cs
  - Validators
    - CountryValidator.cs
    - OwnerValidator.cs
    - PortValidator.cs
    - VesselTypeValidator.cs
    - VesselValidator.cs
  - .genesis

1) **DBContext**
2) Models
3) Repositories
4) Validators

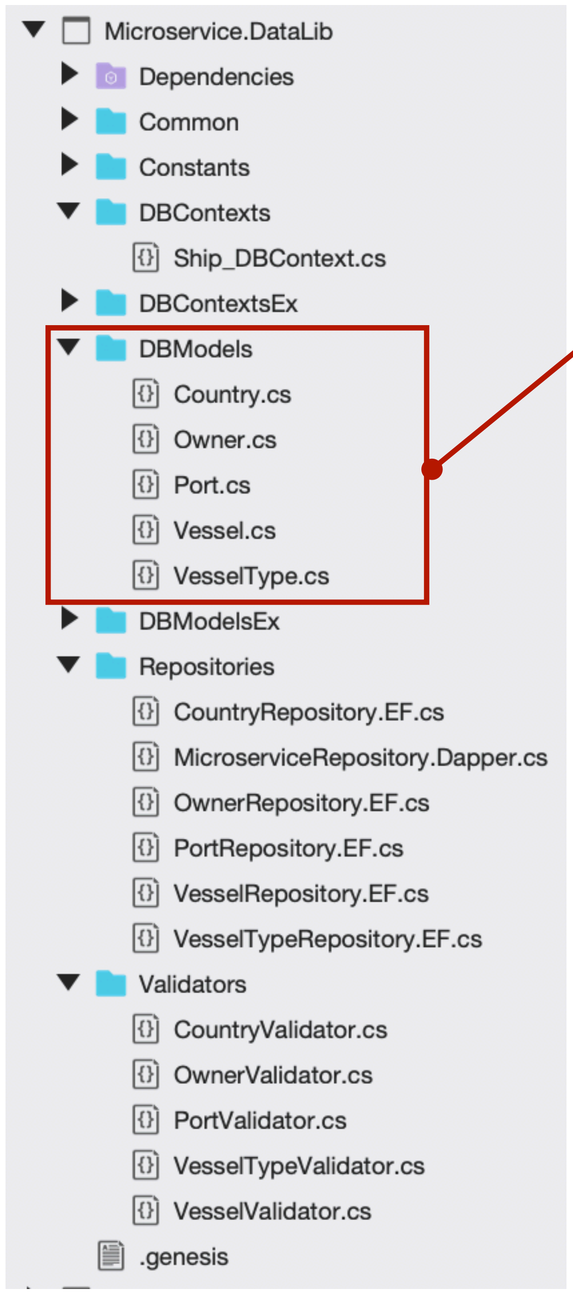
Extended and/or partial classes



```
public virtual DbSet<Country> Country { get; set; }
public virtual DbSet<Owner> Owner { get; set; }
public virtual DbSet<Port> Port { get; set; }
public virtual DbSet<Vessel> Vessel { get; set; }
public virtual DbSet<VesselType> VesselType { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
    base.OnConfiguring(optionsBuilder);

    if (!optionsBuilder.IsConfigured)
    {
        optionsBuilder.UseNpgsql(ConfigurationManager.GetConnectionString("PostgreSQL"), b => b.MigrationsAssembly("Microservice.API"));
    }
}

protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    base.OnModelCreating(modelBuilder);

    modelBuilder.HasAnnotation("ProductVersion", "2.2.6-servicing-10079");

    modelBuilder.Entity<Country>(entity =>
    {
        entity.Property(e => e.OrderNumber).HasDefaultValueSql("0");
    });

    modelBuilder.Entity<Port>(entity =>
    {
        entity.HasOne(d => d.CountryNavigation)
            .WithMany(p => p.Port)
            .HasForeignKey(d => d.Country)
            .HasConstraintName("port_fk");
    });

    modelBuilder.Entity<Vessel>(entity =>
    {
        entity.HasOne(d => d.FlagNavigation)
            .WithMany(p => p.Vessel)
            .HasForeignKey(d => d.Flag)
            .OnDelete(DeleteBehavior.ClientSetNull)
            .HasConstraintName("vessel_country_fk");

        entity.HasOne(d => d.HomePortNavigation)
            .WithMany(p => p.Vessel)
            .HasForeignKey(d => d.HomePort)
            .OnDelete(DeleteBehavior.ClientSetNull)
            .HasConstraintName("vessel_port_fk");

        entity.HasOne(d => d.OwnerNavigation)
            .WithMany(p => p.Vessel)
            .HasForeignKey(d => d.Owner)
            .OnDelete(DeleteBehavior.ClientSetNull)
```

Generated Backend / Data Layer

2) DB Models

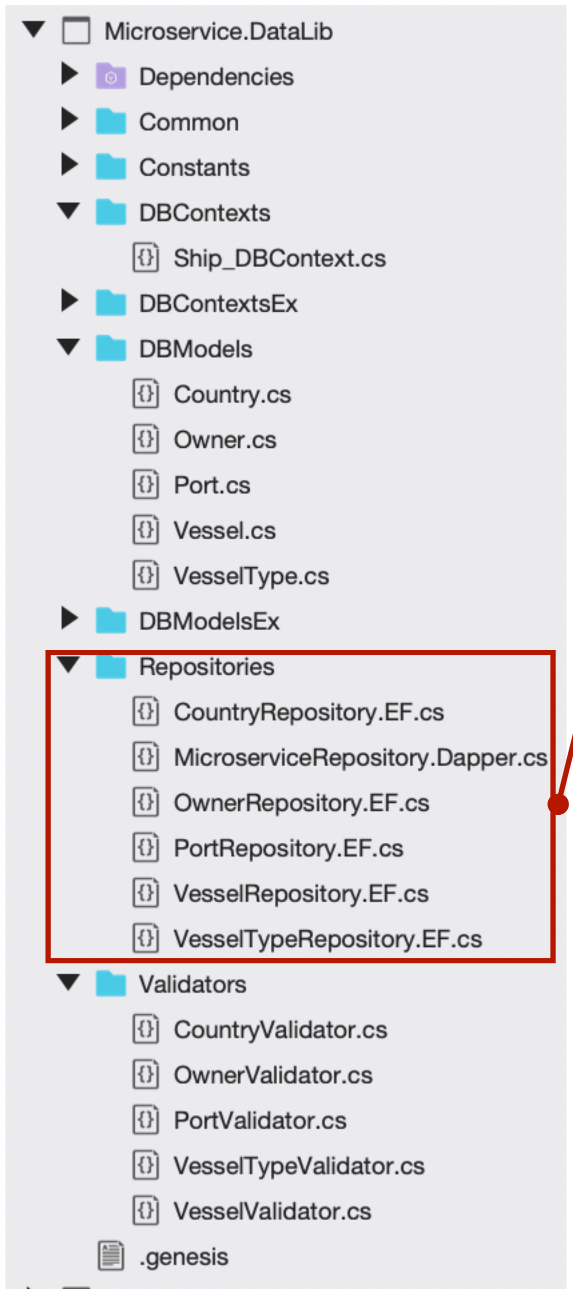


1) DBContext
2) Models
3) Repositories
4) Validators

```
namespace Microservice.DataLib.DBModels
{
    [Table("vessel")]
    public partial class Vessel
    {
        [Column("vesselId")]
        public int VesselId { get; set; }
        [Required]
        [StringLength(10)]
        public string ImoNumber { get; set; }
        [Required]
        [Column("vesselName")]
        [StringLength(20)]
        public string VesselName { get; set; }
        [Column("owner")]
        public int Owner { get; set; }
        [Column("homePort")]
        public int HomePort { get; set; }
        [Column("flag")]
        public int Flag { get; set; }
        [Column("grossTonnage")]
        public long? GrossTonnage { get; set; }
        [Column("yearOfBuilt")]
        public int? YearOfBuilt { get; set; }
        [Column("lengthOverall")]
        public int? LengthOverall { get; set; }
        [Column("maxSpeed", TypeName = "numeric(5,2)")]
        public decimal? MaxSpeed { get; set; }
        [Column("description")]
        public string Description { get; set; }
        [Column("vesselPicture")]
        public string VesselPicture { get; set; }
        [Column("vesselType")]
        public int VesselType { get; set; }
        [Column("lastTrip", TypeName = "date")]
        public DateTime? LastTrip { get; set; }

        [ForeignKey("Flag")]
        [InverseProperty("Vessel")]
        public virtual Country FlagNavigation { get; set; }
        [ForeignKey("HomePort")]
        [InverseProperty("Vessel")]
        public virtual Port HomePortNavigation { get; set; }
        [ForeignKey("Owner")]
        [InverseProperty("Vessel")]
        public virtual Owner OwnerNavigation { get; set; }
        [ForeignKey("VesselType")]
        [InverseProperty("Vessel")]
        public virtual VesselType VesselTypeNavigation { get; set; }
    }
}
```

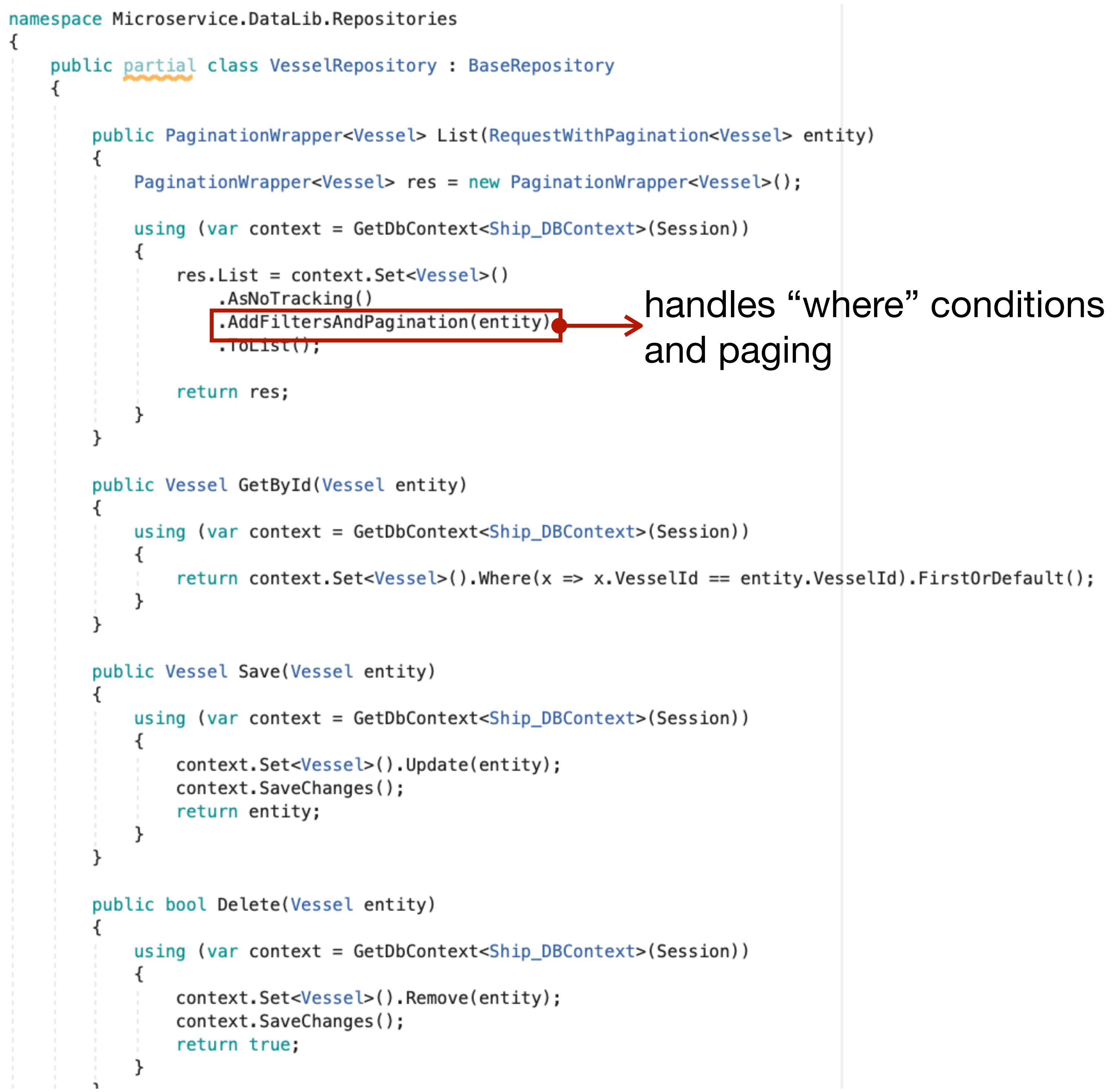
Generated Backend / Data Layer

3) Repositories



- Microservice.DataLib
  - Dependencies
  - Common
  - Constants
  - DBContexts
    - Ship_DBContext.cs
  - DBContextsEx
  - DBModels
    - Country.cs
    - Owner.cs
    - Port.cs
    - Vessel.cs
    - VesselType.cs
  - DBModelsEx
  - Repositories
    - CountryRepository.EF.cs
    - MicroserviceRepository.Dapper.cs
    - OwnerRepository.EF.cs
    - PortRepository.EF.cs
    - VesselRepository.EF.cs
    - VesselTypeRepository.EF.cs
  - Validators
    - CountryValidator.cs
    - OwnerValidator.cs
    - PortValidator.cs
    - VesselTypeValidator.cs
    - VesselValidator.cs
  - .genesis

1) DBContext
2) Models
3) **Repositories**
4) Validators



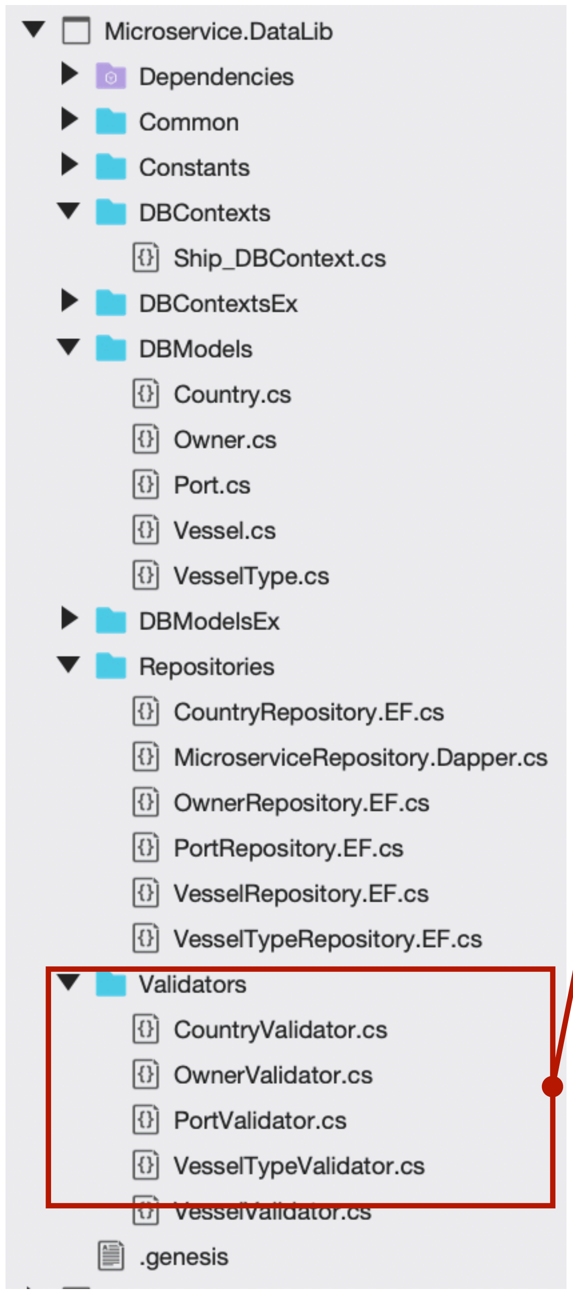
```
namespace Microservice.DataLib.Repositories
{
    public partial class VesselRepository : BaseRepository
    {
        public PaginationWrapper<Vessel> List(RequestWithPagination<Vessel> entity)
        {
            PaginationWrapper<Vessel> res = new PaginationWrapper<Vessel>();

            using (var context = GetDbContext<Ship_DBContext>(Session))
            {
                res.List = context.Set<Vessel>()
                    .AsNoTracking()
                    .AddFiltersAndPagination(entity)
                    .ToList();

                return res;
            }
        }

        public Vessel GetById(Vessel entity)
        {
            using (var context = GetDbContext<Ship_DBContext>(Session))
            {
                return context.Set<Vessel>().Where(x => x.VesselId == entity.VesselId).FirstOrDefault();
            }
        }

        public Vessel Save(Vessel entity)
        {
            using (var context = GetDbContext<Ship_DBContext>(Session))
            {
                context.Set<Vessel>().Update(entity);
                context.SaveChanges();
                return entity;
            }
        }

        public bool Delete(Vessel entity)
        {
            using (var context = GetDbContext<Ship_DBContext>(Session))
            {
                context.Set<Vessel>().Remove(entity);
                context.SaveChanges();
                return true;
            }
```

handles “where” conditions and paging

NET CORE GENESIS

Generated Backend / Data Layer

4) Validators *(FluentValidation)*



1) DBContext
2) Models
3) Repositories
4) **Validators**

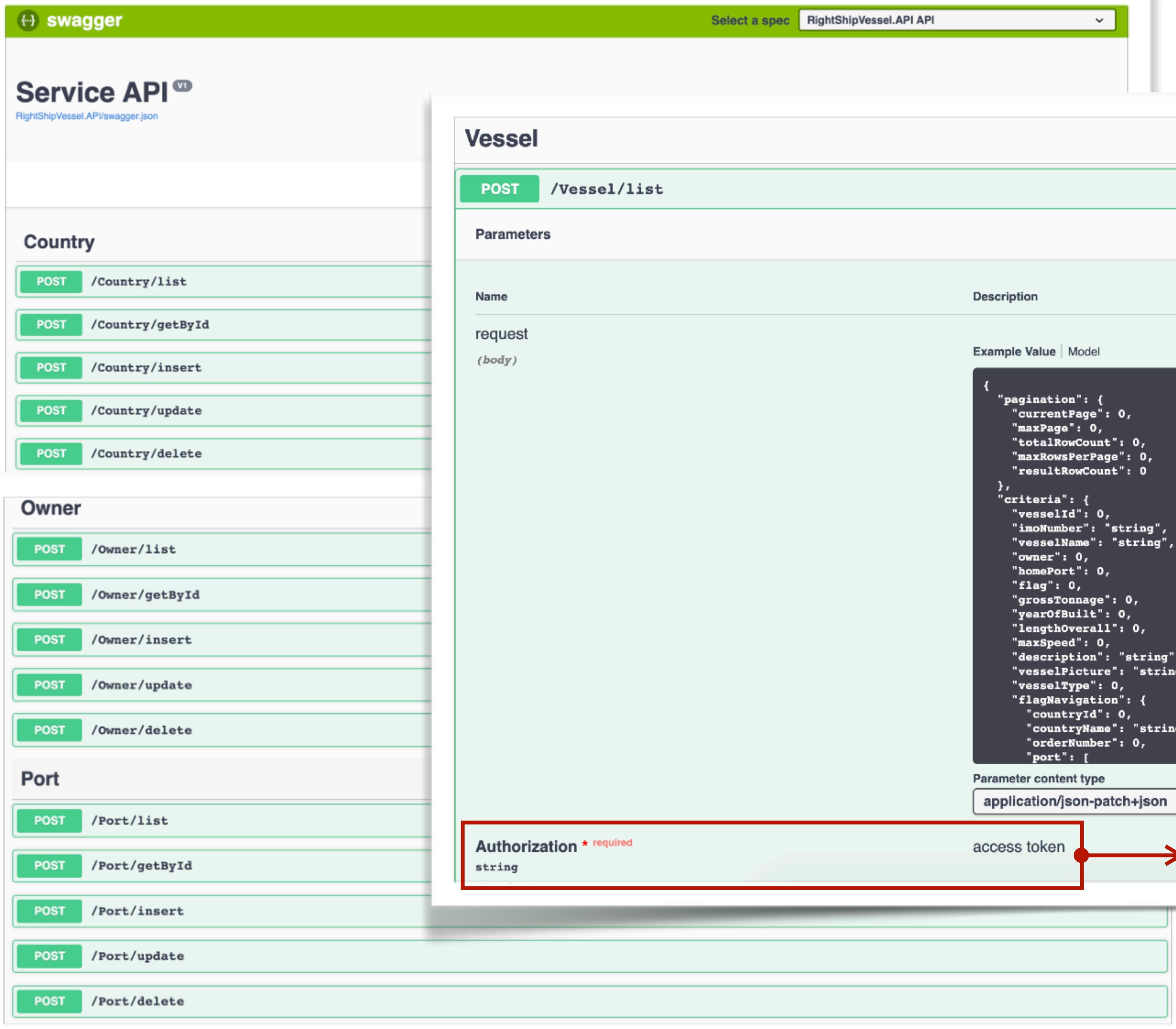
```
namespace Microservice.DataLib.Validators
{
    public class VesselValidator : AbstractValidator<Vessel>
    {
        public VesselValidator()
        {
            RuleFor(x => x.VesselId)
                .NotNull();

            RuleFor(x => x.ImoNumber)
                .NotNull()
                .MaximumLength(10);

            RuleFor(x => x.VesselName)
                .NotNull()
                .MaximumLength(20);

            RuleFor(x => x.Owner)
                .NotNull();

            RuleFor(x => x.HomePort)
                .NotNull();

            RuleFor(x => x.Flag)
                .NotNull();

            RuleFor(x => x.VesselType)
                .NotNull();
        }
    }
}
```

NET CORE GENESIS

Backend API Layer

API Layer and Web Services are ready-to-use

* *Swagger*
* *Open-API 3.0 Compliant*



swagger — Select a spec: RightShipVessel.API API

Service API V1

RightShipVessel.API/swagger.json

Country

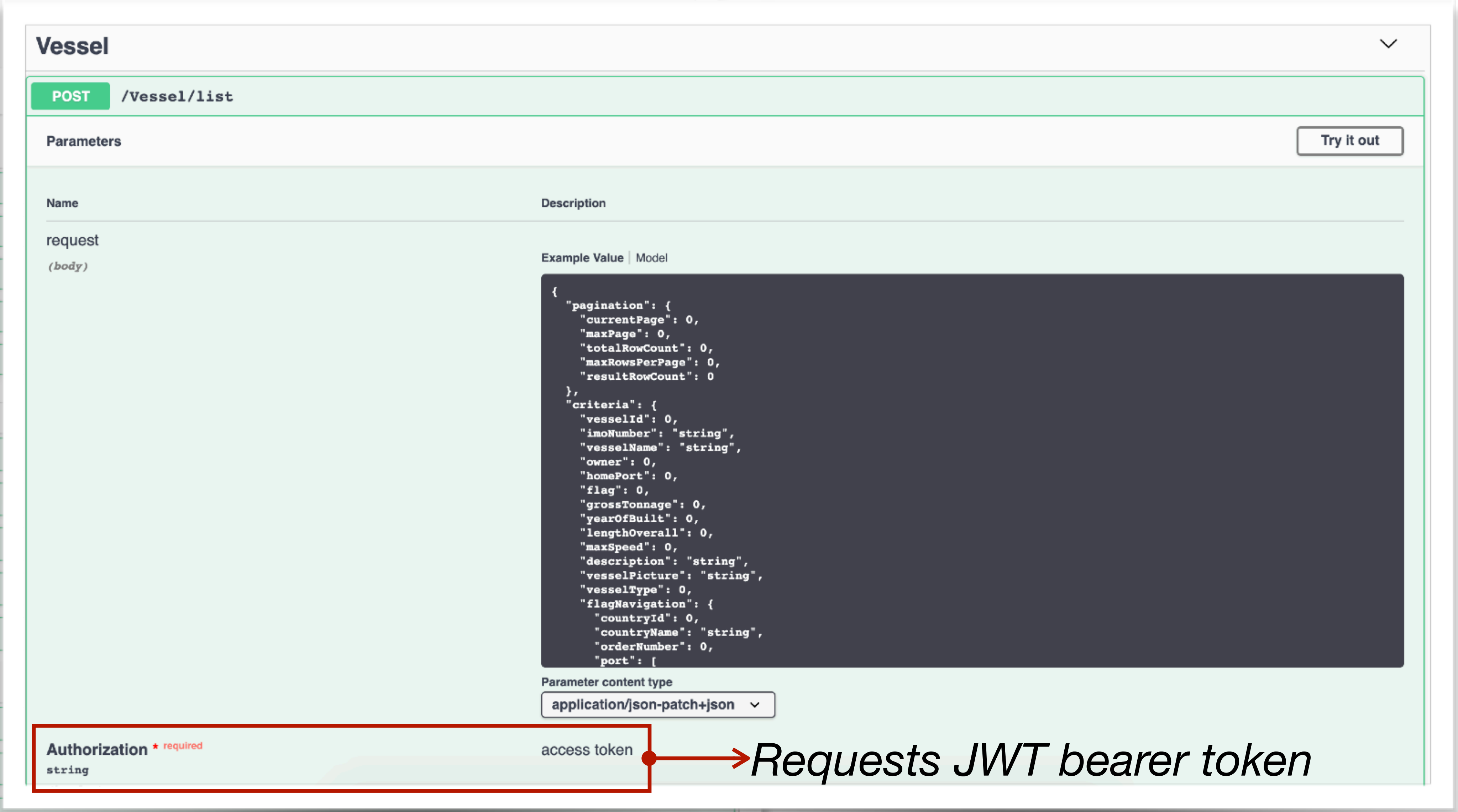
- POST /Country/list
- POST /Country/getById
- POST /Country/insert
- POST /Country/update
- POST /Country/delete

Owner

- POST /Owner/list
- POST /Owner/getById
- POST /Owner/insert
- POST /Owner/update
- POST /Owner/delete

Port

- POST /Port/list
- POST /Port/getById
- POST /Port/insert
- POST /Port/update
- POST /Port/delete



Vessel

POST /Vessel/list

Parameters — Try it out

Name	Description
request *(body)*	Example Value | Model

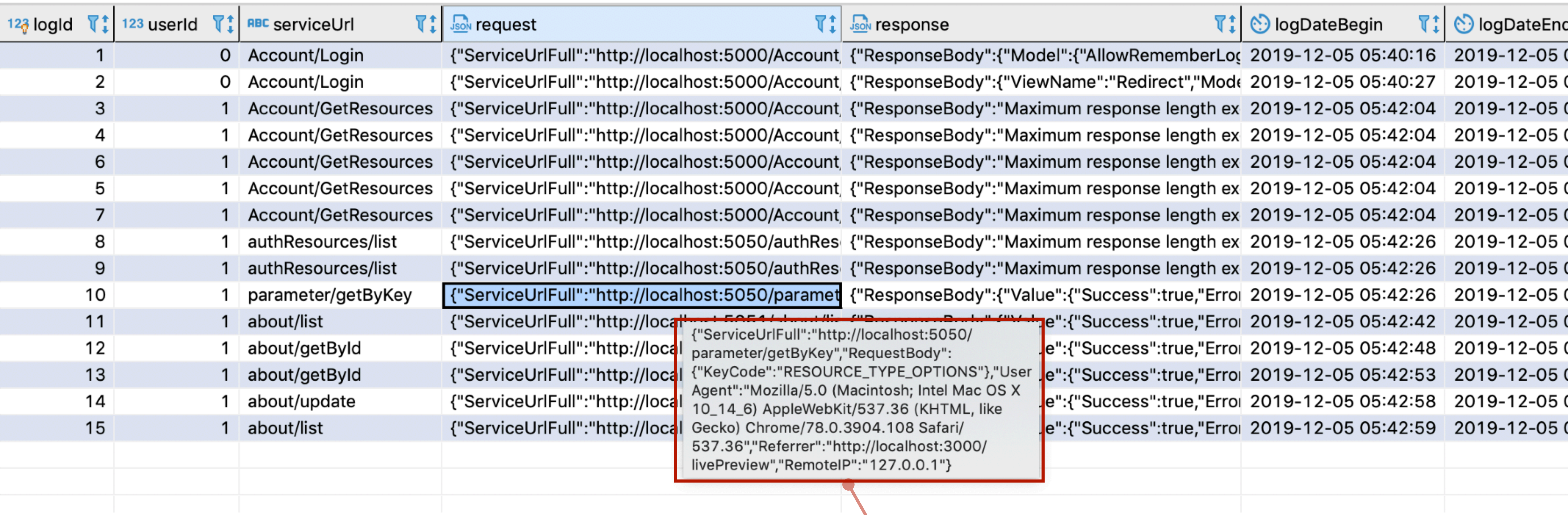
```
{
  "pagination": {
    "currentPage": 0,
    "maxPage": 0,
    "totalRowCount": 0,
    "maxRowsPerPage": 0,
    "resultRowCount": 0
  },
  "criteria": {
    "vesselId": 0,
    "imoNumber": "string",
    "vesselName": "string",
    "owner": 0,
    "homePort": 0,
    "flag": 0,
    "grossTonnage": 0,
    "yearOfBuilt": 0,
    "lengthOverall": 0,
    "maxSpeed": 0,
    "description": "string",
    "vesselPicture": "string",
    "vesselType": 0,
    "flagNavigation": {
      "countryId": 0,
      "countryName": "string",
      "orderNumber": 0,
      "port": [
```

Parameter content type: application/json-patch+json

Authorization * required string	access token

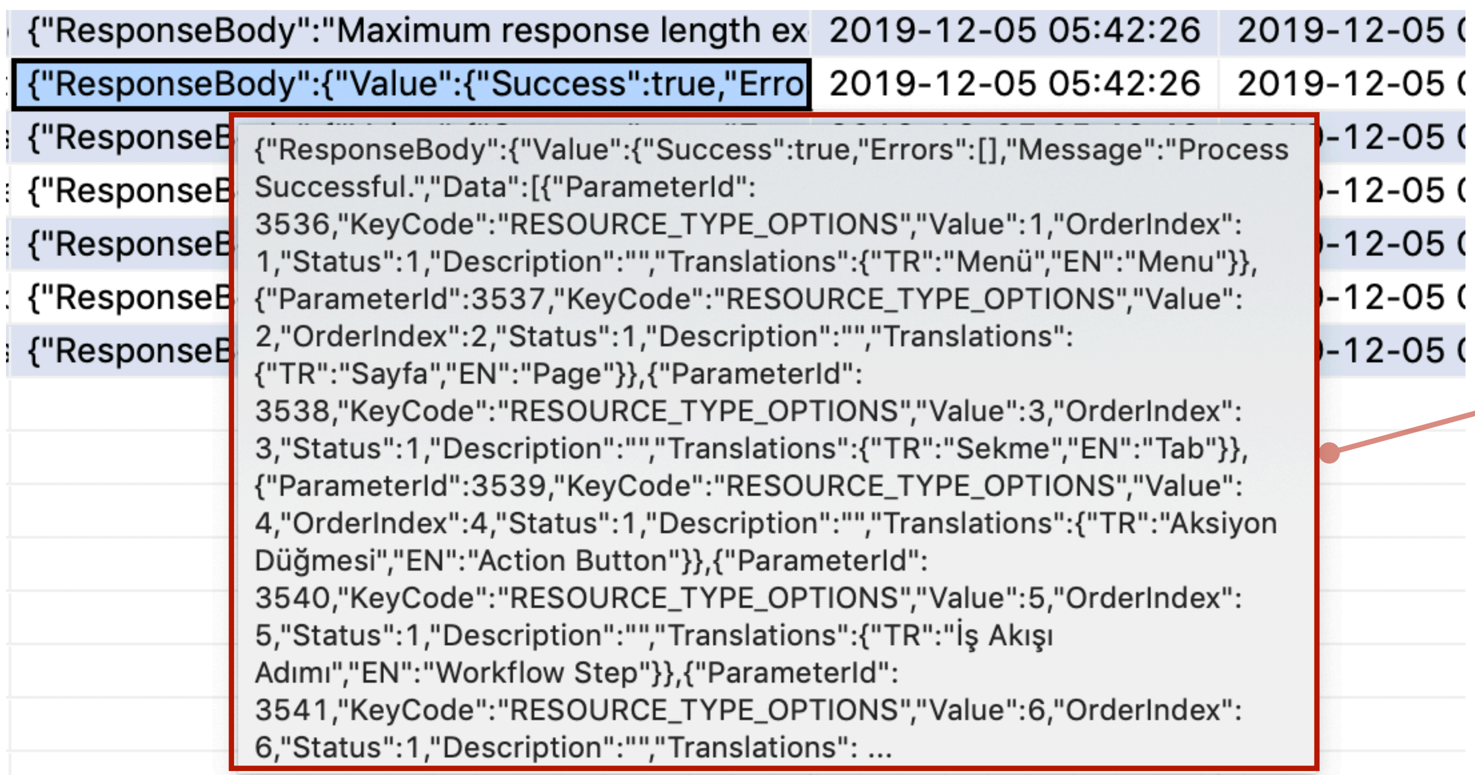
Requests JWT bearer token

Logging Middleware



logId	userId	serviceUrl	request	response	logDateBegin	logDateEnd
1	0	Account/Login	{"ServiceUrlFull":"http://localhost:5000/Account	{"ResponseBody":{"Model":{"AllowRememberLog	2019-12-05 05:40:16	2019-12-05 0
2	0	Account/Login	{"ServiceUrlFull":"http://localhost:5000/Account	{"ResponseBody":{"ViewName":"Redirect","Mode	2019-12-05 05:40:27	2019-12-05
3	1	Account/GetResources	{"ServiceUrlFull":"http://localhost:5000/Account	{"ResponseBody":"Maximum response length ex	2019-12-05 05:42:04	2019-12-05 0
4	1	Account/GetResources	{"ServiceUrlFull":"http://localhost:5000/Account	{"ResponseBody":"Maximum response length ex	2019-12-05 05:42:04	2019-12-05
6	1	Account/GetResources	{"ServiceUrlFull":"http://localhost:5000/Account	{"ResponseBody":"Maximum response length ex	2019-12-05 05:42:04	2019-12-05
5	1	Account/GetResources	{"ServiceUrlFull":"http://localhost:5000/Account	{"ResponseBody":"Maximum response length ex	2019-12-05 05:42:04	2019-12-05 0
7	1	Account/GetResources	{"ServiceUrlFull":"http://localhost:5000/Account	{"ResponseBody":"Maximum response length ex	2019-12-05 05:42:04	2019-12-05
8	1	authResources/list	{"ServiceUrlFull":"http://localhost:5050/authRes	{"ResponseBody":"Maximum response length ex	2019-12-05 05:42:26	2019-12-05
9	1	authResources/list	{"ServiceUrlFull":"http://localhost:5050/authRes	{"ResponseBody":"Maximum response length ex	2019-12-05 05:42:26	2019-12-05 0
10	1	parameter/getByKey	{"ServiceUrlFull":"http://localhost:5050/paramet	{"ResponseBody":{"Value":{"Success":true,"Erroi	2019-12-05 05:42:26	2019-12-05
11	1	about/list	{"ServiceUrlFull":"http://loca	e":{"Success":true,"Erroi	2019-12-05 05:42:42	2019-12-05
12	1	about/getById	{"ServiceUrlFull":"http://local	e":{"Success":true,"Erroi	2019-12-05 05:42:48	2019-12-05 0
13	1	about/getById	{"ServiceUrlFull":"http://local	e":{"Success":true,"Erroi	2019-12-05 05:42:53	2019-12-05
14	1	about/update	{"ServiceUrlFull":"http://local	e":{"Success":true,"Erroi	2019-12-05 05:42:58	2019-12-05 0
15	1	about/list	{"ServiceUrlFull":"http://local	e":{"Success":true,"Erroi	2019-12-05 05:42:59	2019-12-05 0

```
{"ServiceUrlFull":"http://localhost:5050/
parameter/getByKey","RequestBody":
{"KeyCode":"RESOURCE_TYPE_OPTIONS"},"User
Agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X
10_14_6) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/78.0.3904.108 Safari/
537.36","Referrer":"http://localhost:3000/
livePreview","RemoteIP":"127.0.0.1"}
```



{"ResponseBody":"Maximum response length ex	2019-12-05 05:42:26	2019-12-05 (
{"ResponseBody":{"Value":{"Success":true,"Erro	2019-12-05 05:42:26	2019-12-05 (
{"ResponseE		-12-05 (
{"ResponseE		-12-05 (
{"ResponseE		-12-05 (
{"ResponseE		-12-05 (
{"ResponseE		-12-05 (

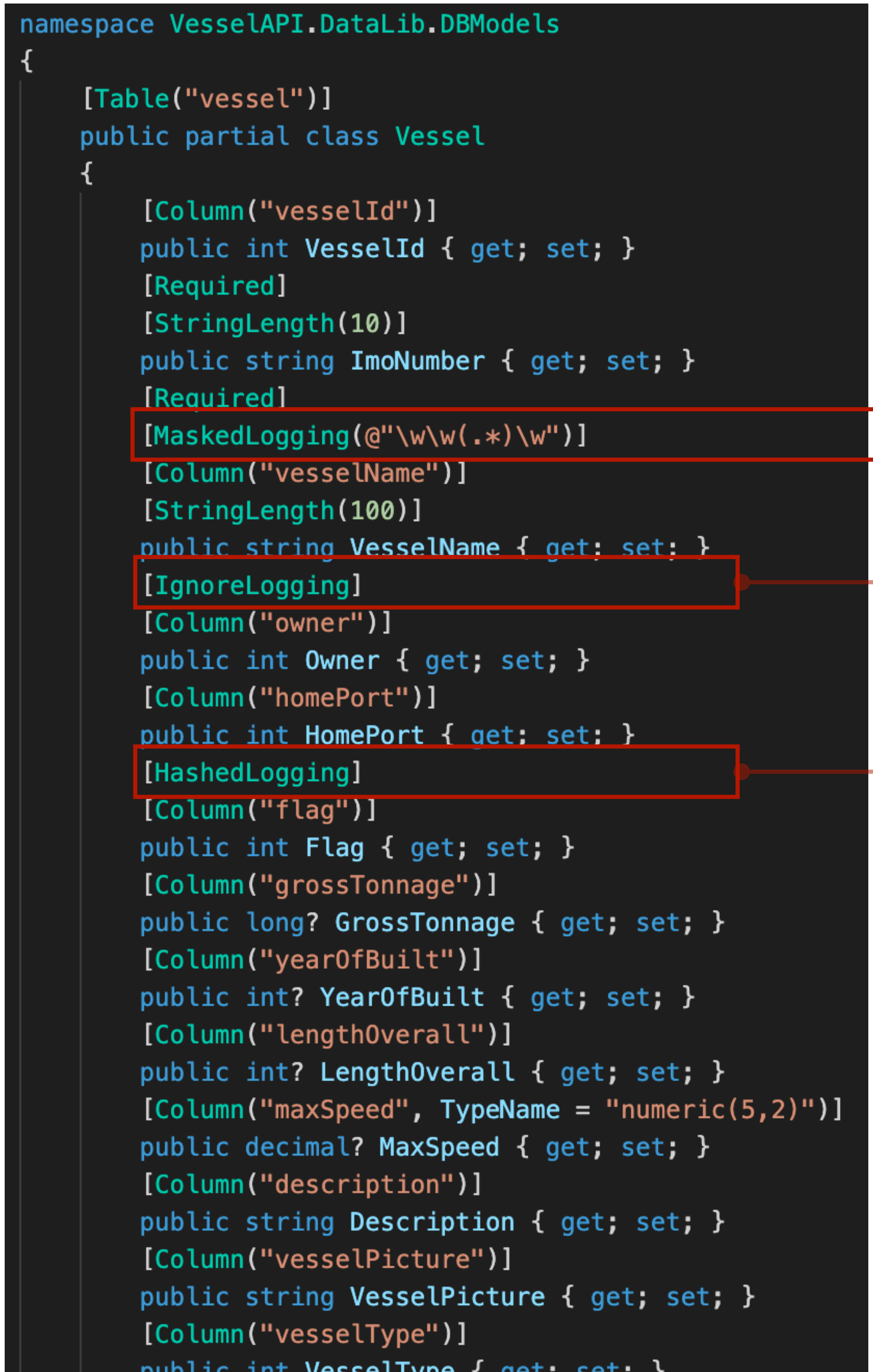
```
{"ResponseBody":{"Value":{"Success":true,"Errors":[],"Message":"Process
Successful.","Data":[{"ParameterId":
3536,"KeyCode":"RESOURCE_TYPE_OPTIONS","Value":1,"OrderIndex":
1,"Status":1,"Description":"","Translations":{"TR":"Menü","EN":"Menu"}},
{"ParameterId":3537,"KeyCode":"RESOURCE_TYPE_OPTIONS","Value":
2,"OrderIndex":2,"Status":1,"Description":"","Translations":
{"TR":"Sayfa","EN":"Page"}},{"ParameterId":
3538,"KeyCode":"RESOURCE_TYPE_OPTIONS","Value":3,"OrderIndex":
3,"Status":1,"Description":"","Translations":{"TR":"Sekme","EN":"Tab"}},
{"ParameterId":3539,"KeyCode":"RESOURCE_TYPE_OPTIONS","Value":
4,"OrderIndex":4,"Status":1,"Description":"","Translations":{"TR":"Aksiyon
Düğmesi","EN":"Action Button"}},{"ParameterId":
3540,"KeyCode":"RESOURCE_TYPE_OPTIONS","Value":5,"OrderIndex":
5,"Status":1,"Description":"","Translations":{"TR":"İş Akışı
Adımı","EN":"Workflow Step"}},{"ParameterId":
3541,"KeyCode":"RESOURCE_TYPE_OPTIONS","Value":6,"OrderIndex":
6,"Status":1,"Description":"","Translations": ...
```

Every Request & Response pair is logged at once in the same row as json

* *User's current permissions are logged also*

NET CORE GENESIS

Logging Middleware



```
namespace VesselAPI.DataLib.DBModels
{
    [Table("vessel")]
    public partial class Vessel
    {
        [Column("vesselId")]
        public int VesselId { get; set; }
        [Required]
        [StringLength(10)]
        public string ImoNumber { get; set; }
        [Required]
        [MaskedLogging(@"\w\w(.*)\w")]
        [Column("vesselName")]
        [StringLength(100)]
        public string VesselName { get; set; }
        [IgnoreLogging]
        [Column("owner")]
        public int Owner { get; set; }
        [Column("homePort")]
        public int HomePort { get; set; }
        [HashedLogging]
        [Column("flag")]
        public int Flag { get; set; }
        [Column("grossTonnage")]
        public long? GrossTonnage { get; set; }
        [Column("yearOfBuilt")]
        public int? YearOfBuilt { get; set; }
        [Column("lengthOverall")]
        public int? LengthOverall { get; set; }
        [Column("maxSpeed", TypeName = "numeric(5,2)")]
        public decimal? MaxSpeed { get; set; }
        [Column("description")]
        public string Description { get; set; }
        [Column("vesselPicture")]
        public string VesselPicture { get; set; }
        [Column("vesselType")]
        public int VesselType { get; set; }
```

We always seek ways

to simplify & lessen your coding effort

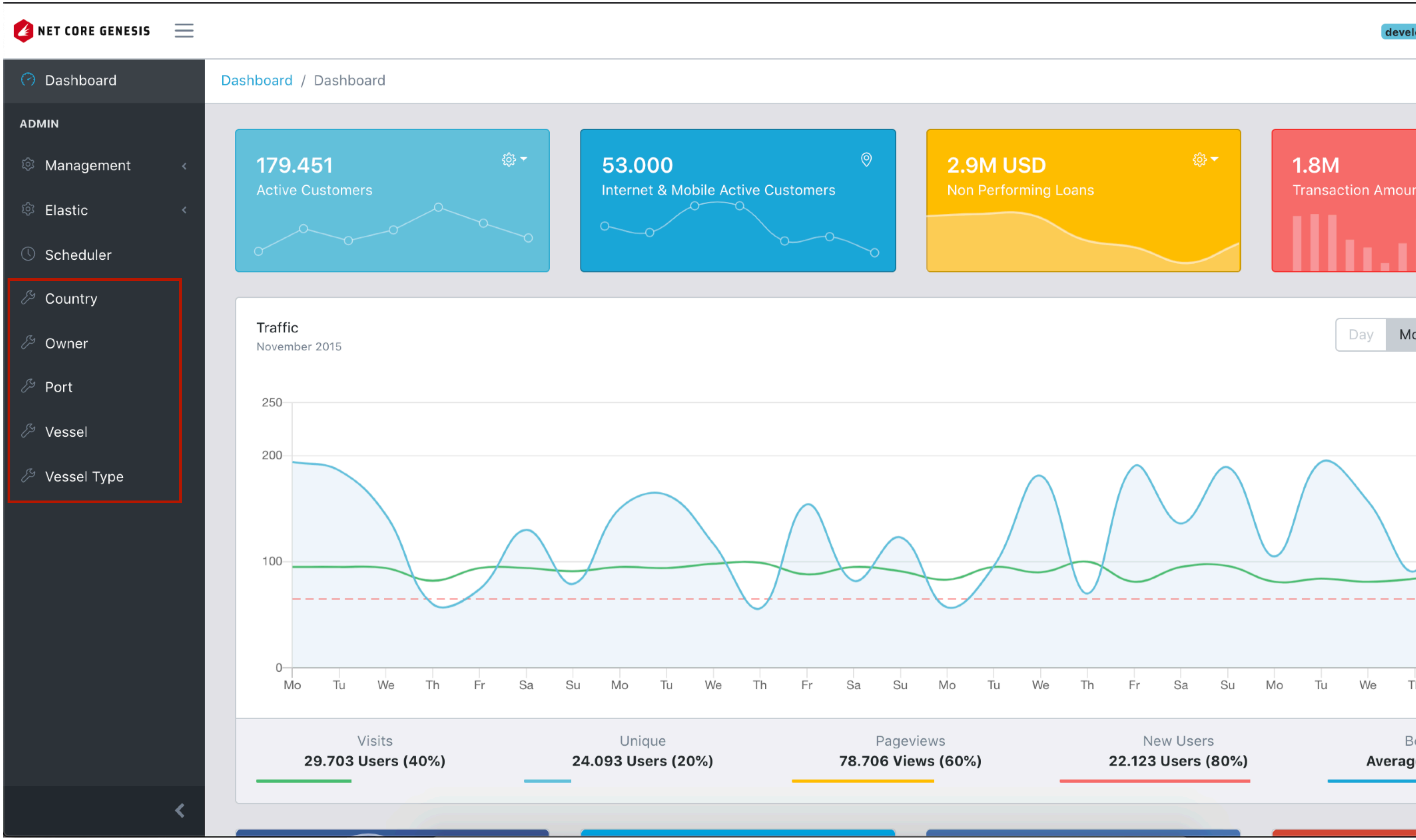
Log the value of “vesselName” as masked

Do not log “owner” property ever

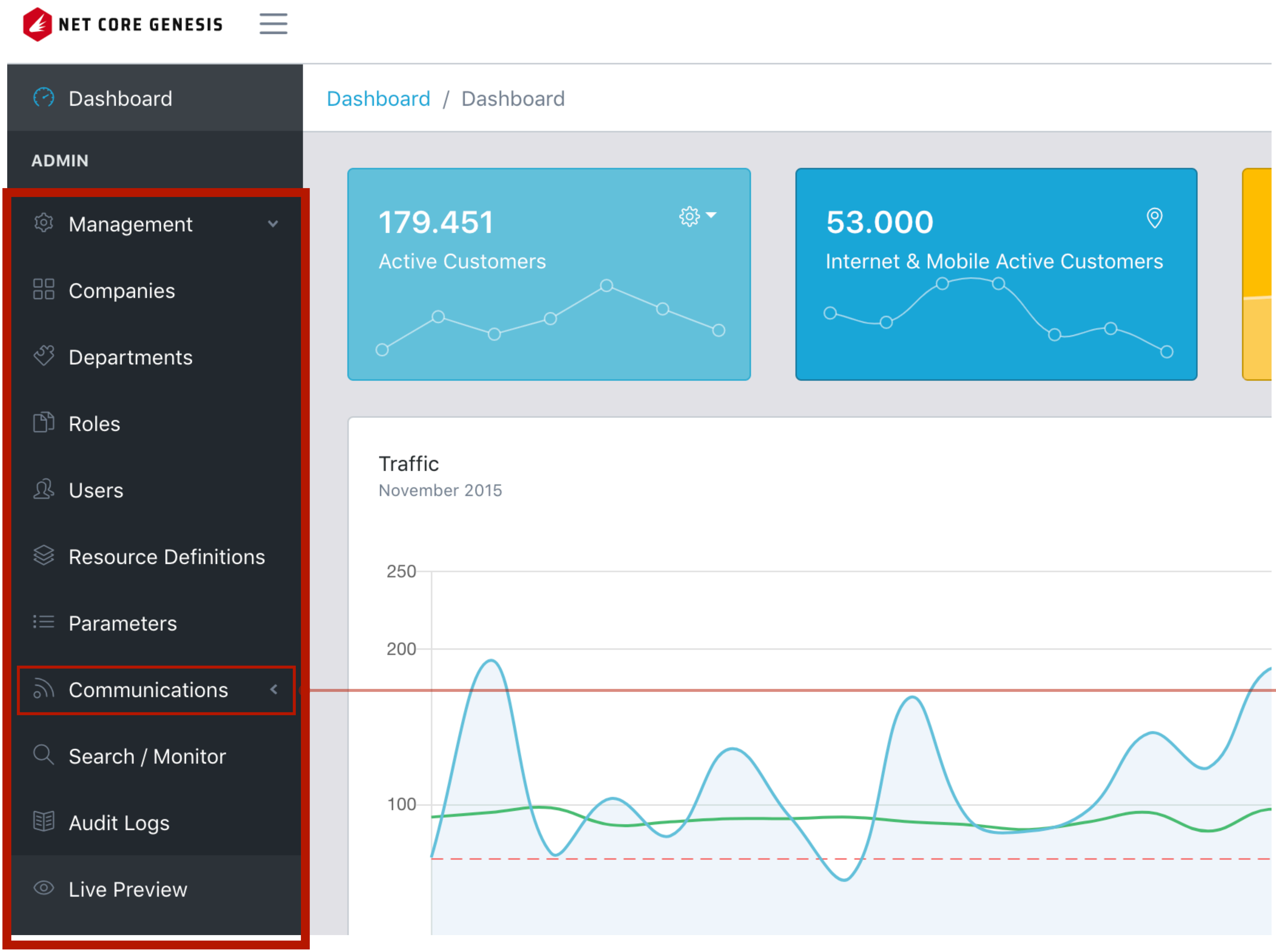
Log the value of “flag” as hashed

NET CORE GENESIS

Frontend / Generated menu items



Frontend / Management Items



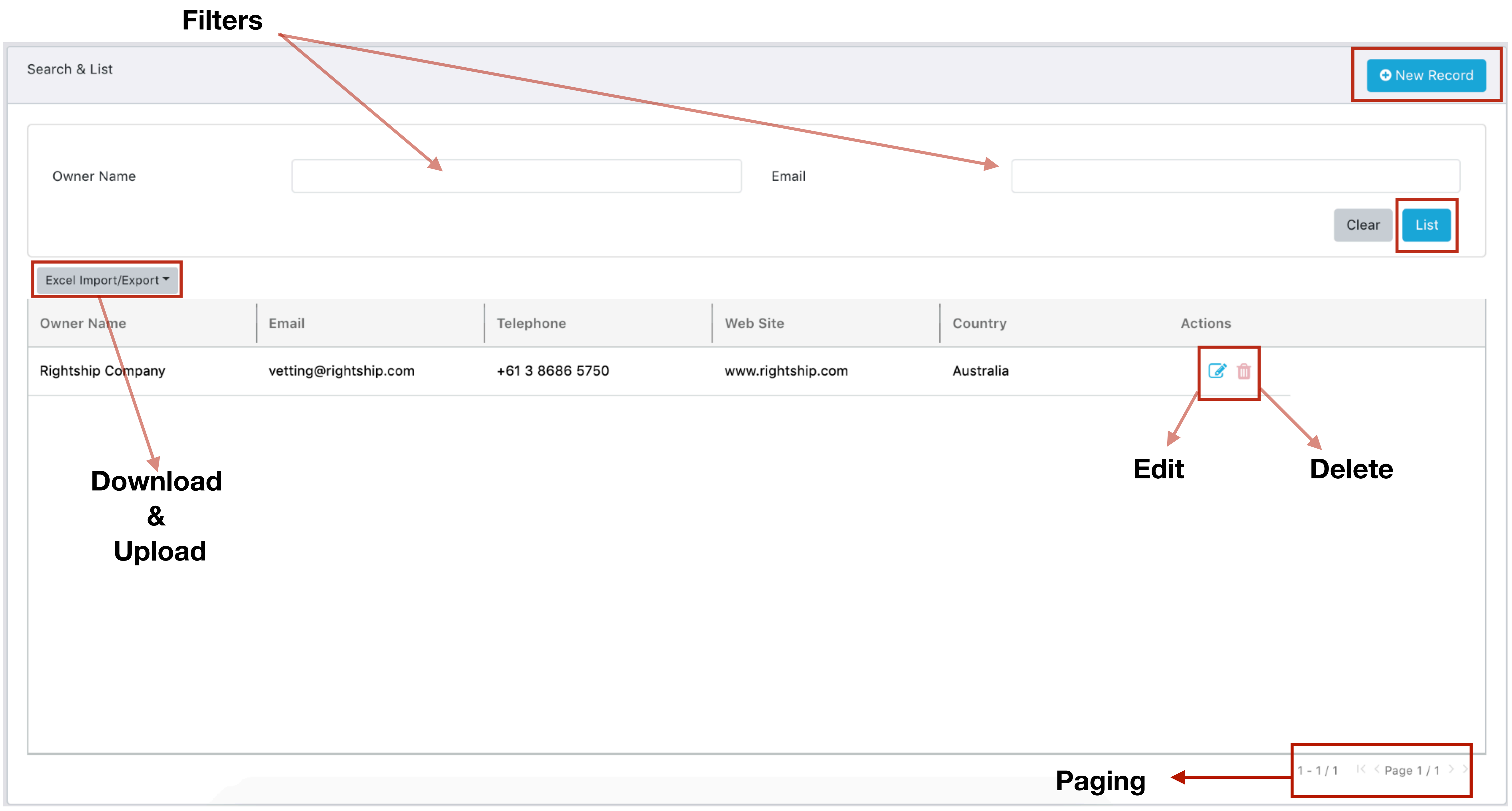
Communication middleware supporting Mail & SMS

NET CORE GENESIS

Frontend / Generated Page

This is the raw view of “Owner” page after creation.

It is fully functional: you can insert, update, delete, get or list



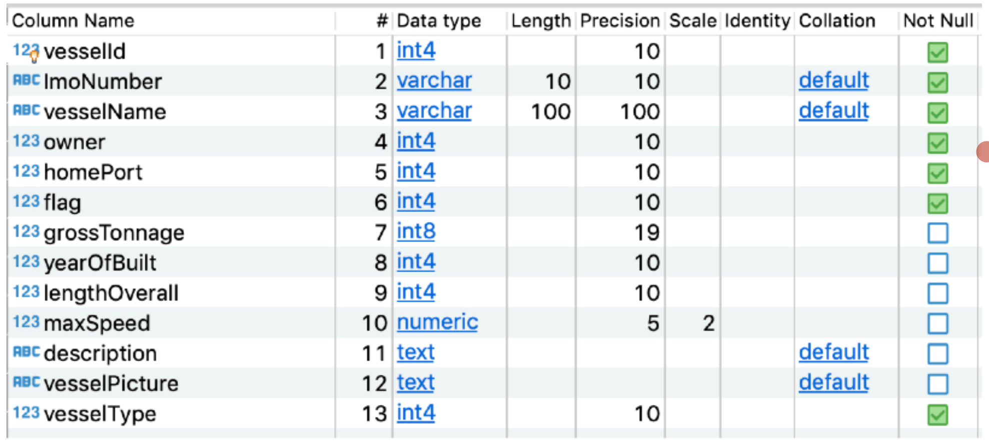
Owner Name	Email	Telephone	Web Site	Country	Actions
Rightship Company	vetting@rightship.com	+61 3 8686 5750	www.rightship.com	Australia	

NET CORE GENESIS

Frontend / JSON based render

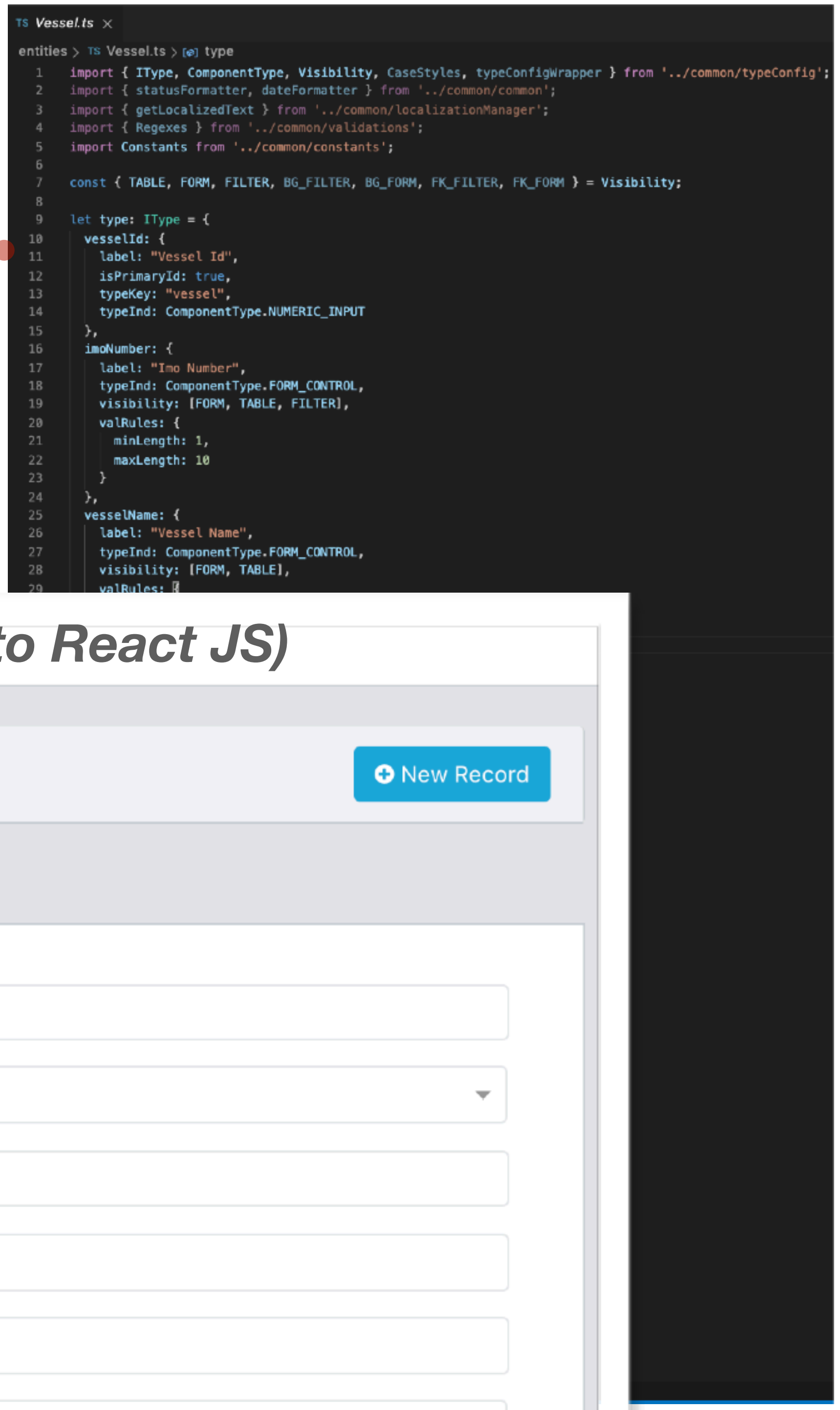
If you prefer; there is a simple, human-readable JSON format to render/change UI screens

(1) From Table



Column Name	#	Data type	Length	Precision	Scale	Identity	Collation	Not Null
vesselId	1	int4		10				✓
ImoNumber	2	varchar	10	10			default	✓
vesselName	3	varchar	100	100			default	✓
owner	4	int4		10				✓
homePort	5	int4		10				✓
flag	6	int4		10				✓
grossTonnage	7	int8		19				☐
yearOfBuilt	8	int4		10				☐
lengthOverall	9	int4		10				☐
maxSpeed	10	numeric		5	2			☐
description	11	text					default	☐
vesselPicture	12	text					default	☐
vesselType	13	int4		10				✓

(2) To JSON

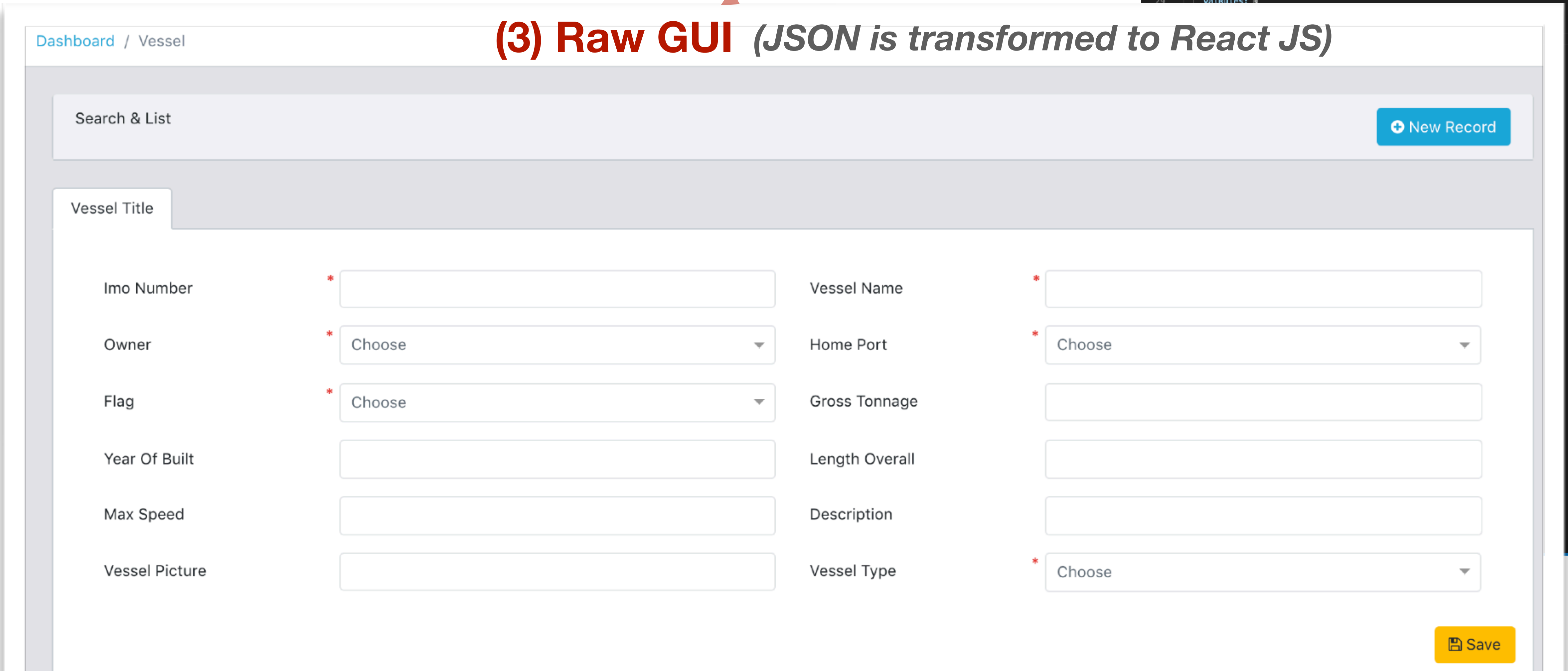


Vessel.ts

entities > Vessel.ts > type

```
import { IType, ComponentType, Visibility, CaseStyles, typeConfigWrapper } from '../common/typeConfig';
import { statusFormatter, dateFormatter } from '../common/common';
import { getLocalizedText } from '../common/localizationManager';
import { Regexes } from '../common/validations';
import Constants from '../common/constants';

const { TABLE, FORM, FILTER, BG_FILTER, BG_FORM, FK_FILTER, FK_FORM } = Visibility;

let type: IType = {
  vesselId: {
    label: "Vessel Id",
    isPrimaryId: true,
    typeKey: "vessel",
    typeInd: ComponentType.NUMERIC_INPUT
  },
  imoNumber: {
    label: "Imo Number",
    typeInd: ComponentType.FORM_CONTROL,
    visibility: [FORM, TABLE, FILTER],
    valRules: {
      minLength: 1,
      maxLength: 10
    }
  },
  vesselName: {
    label: "Vessel Name",
    typeInd: ComponentType.FORM_CONTROL,
    visibility: [FORM, TABLE],
    valRules: {
```

(3) Raw GUI *(JSON is transformed to React JS)*



Dashboard / Vessel

Search & List

New Record

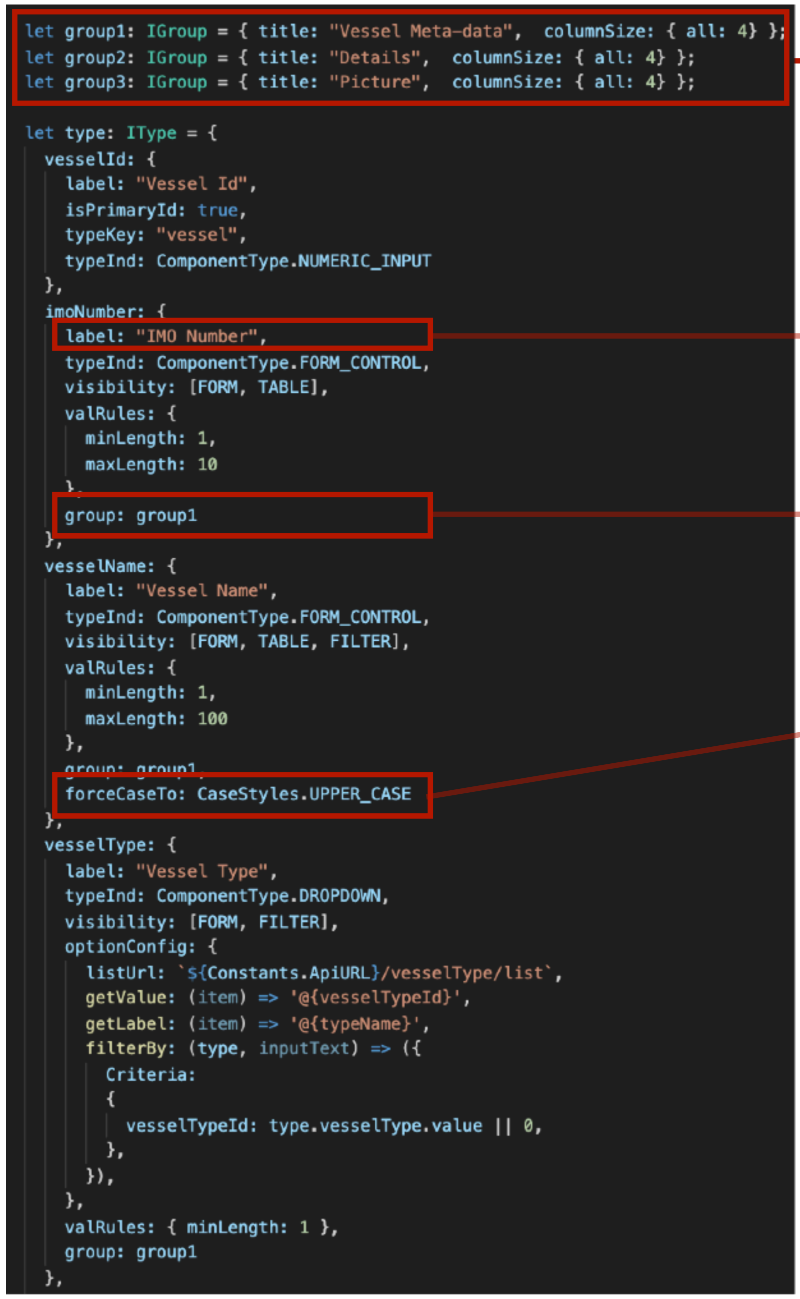
Vessel Title

Imo Number *		Vessel Name *	
Owner *	Choose	Home Port *	Choose
Flag *	Choose	Gross Tonnage	
Year Of Built		Length Overall	
Max Speed		Description	
Vessel Picture		Vessel Type *	Choose

Save

Frontend / Page Edit

Let's make
some fast changes
to Vessel *(even w/o knowing React JS)*



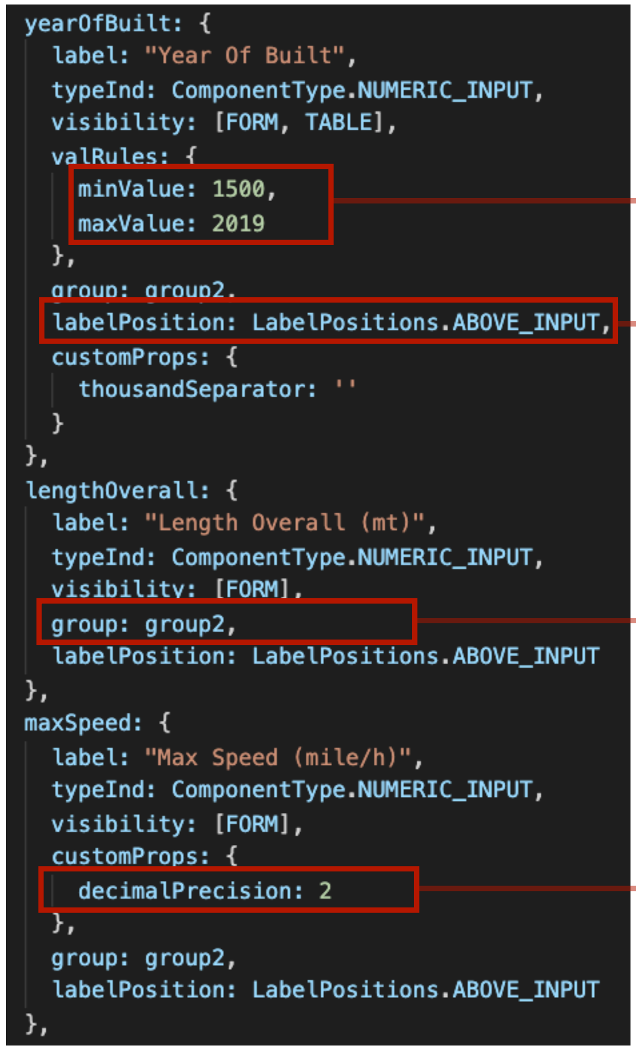
```
let group1: IGroup = { title: "Vessel Meta-data",  columnSize: { all: 4} };
let group2: IGroup = { title: "Details",  columnSize: { all: 4} };
let group3: IGroup = { title: "Picture",  columnSize: { all: 4} };

let type: IType = {
  vesselId: {
    label: "Vessel Id",
    isPrimaryId: true,
    typeKey: "vessel",
    typeInd: ComponentType.NUMERIC_INPUT
  },
  imoNumber: {
    label: "IMO Number",
    typeInd: ComponentType.FORM_CONTROL,
    visibility: [FORM, TABLE],
    valRules: {
      minLength: 1,
      maxLength: 10
    },
    group: group1
  },
  vesselName: {
    label: "Vessel Name",
    typeInd: ComponentType.FORM_CONTROL,
    visibility: [FORM, TABLE, FILTER],
    valRules: {
      minLength: 1,
      maxLength: 100
    },
    group: group1,
    forceCaseTo: CaseStyles.UPPER_CASE
  },
  vesselType: {
    label: "Vessel Type",
    typeInd: ComponentType.DROPDOWN,
    visibility: [FORM, FILTER],
    optionConfig: {
      listUrl: `${Constants.ApiURL}/vesselType/list`,
      getValue: (item) => '@{vesselTypeId}',
      getLabel: (item) => '@{typeName}',
      filterBy: (type, inputText) => ({
        Criteria:
        {
          vesselTypeId: type.vesselType.value || 0,
        },
      }),
    },
    valRules: { minLength: 1 },
    group: group1
  },
```

Define 3 groups

Change label

Place in group1

Make the text
upper case instantly



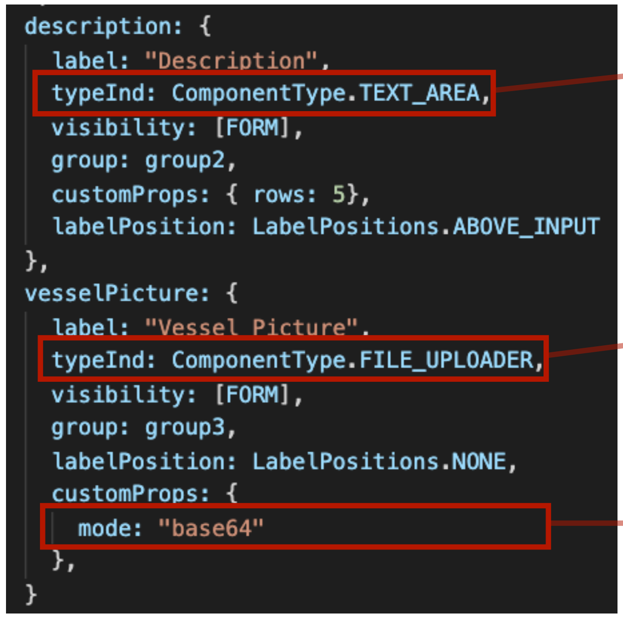
```
yearOfBuilt: {
  label: "Year Of Built",
  typeInd: ComponentType.NUMERIC_INPUT,
  visibility: [FORM, TABLE],
  valRules: {
    minValue: 1500,
    maxValue: 2019
  },
  group: group2,
  labelPosition: LabelPositions.ABOVE_INPUT,
  customProps: {
    thousandSeparator: ''
  }
},
lengthOverall: {
  label: "Length Overall (mt)",
  typeInd: ComponentType.NUMERIC_INPUT,
  visibility: [FORM],
  group: group2,
  labelPosition: LabelPositions.ABOVE_INPUT
},
maxSpeed: {
  label: "Max Speed (mile/h)",
  typeInd: ComponentType.NUMERIC_INPUT,
  visibility: [FORM],
  customProps: {
    decimalPrecision: 2
  },
  group: group2,
  labelPosition: LabelPositions.ABOVE_INPUT
},
```

Set min & max values

Place label above

Place in group2

Set decimal precision



```
description: {
  label: "Description",
  typeInd: ComponentType.TEXT_AREA,
  visibility: [FORM],
  group: group2,
  customProps: { rows: 5},
  labelPosition: LabelPositions.ABOVE_INPUT
},
vesselPicture: {
  label: "Vessel Picture",
  typeInd: ComponentType.FILE_UPLOADER,
  visibility: [FORM],
  group: group3,
  labelPosition: LabelPositions.NONE,
  customProps: {
    mode: "base64"
  },
}
```

Component type
to Text Area

Component type
to File Uploader

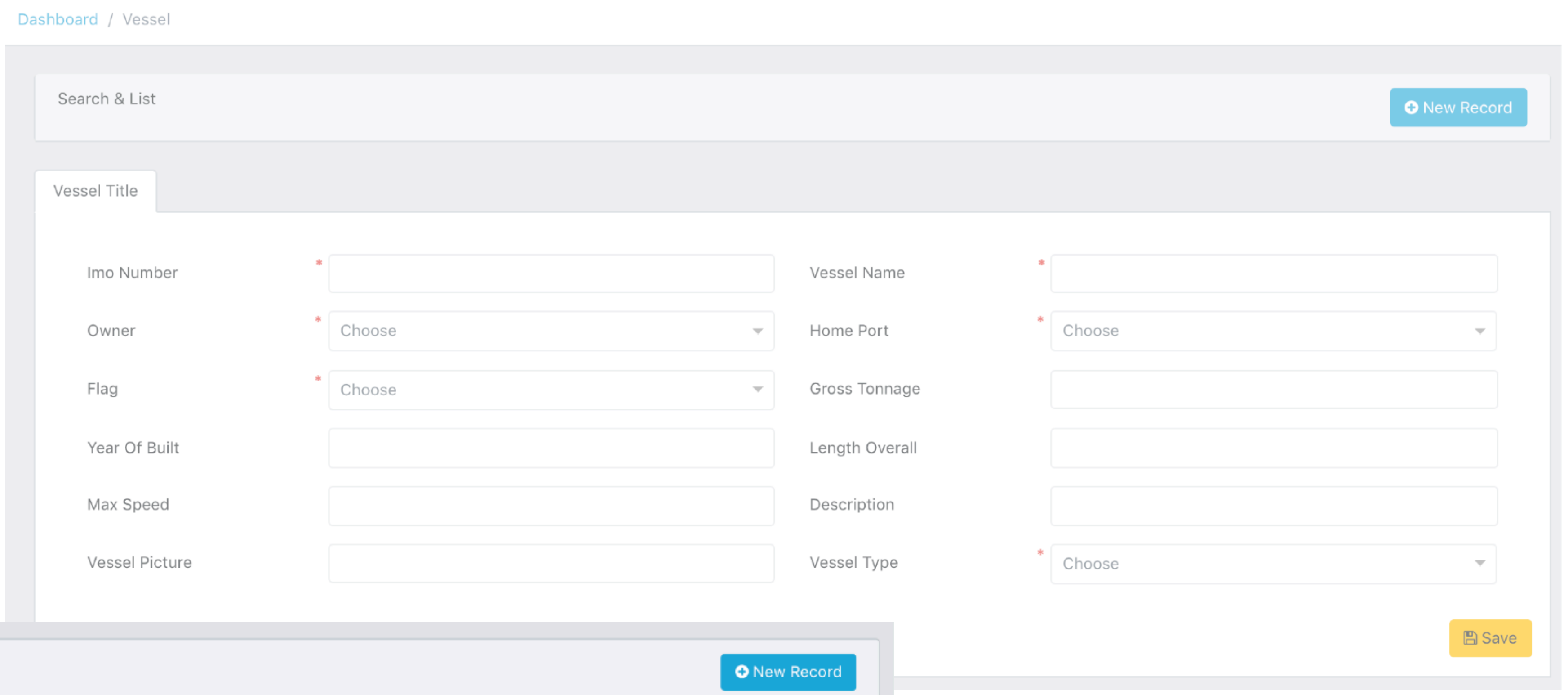
Send value as base64

NET CORE GENESIS

Frontend / The magic happens

In 2 (two) minutes, without coding

Before



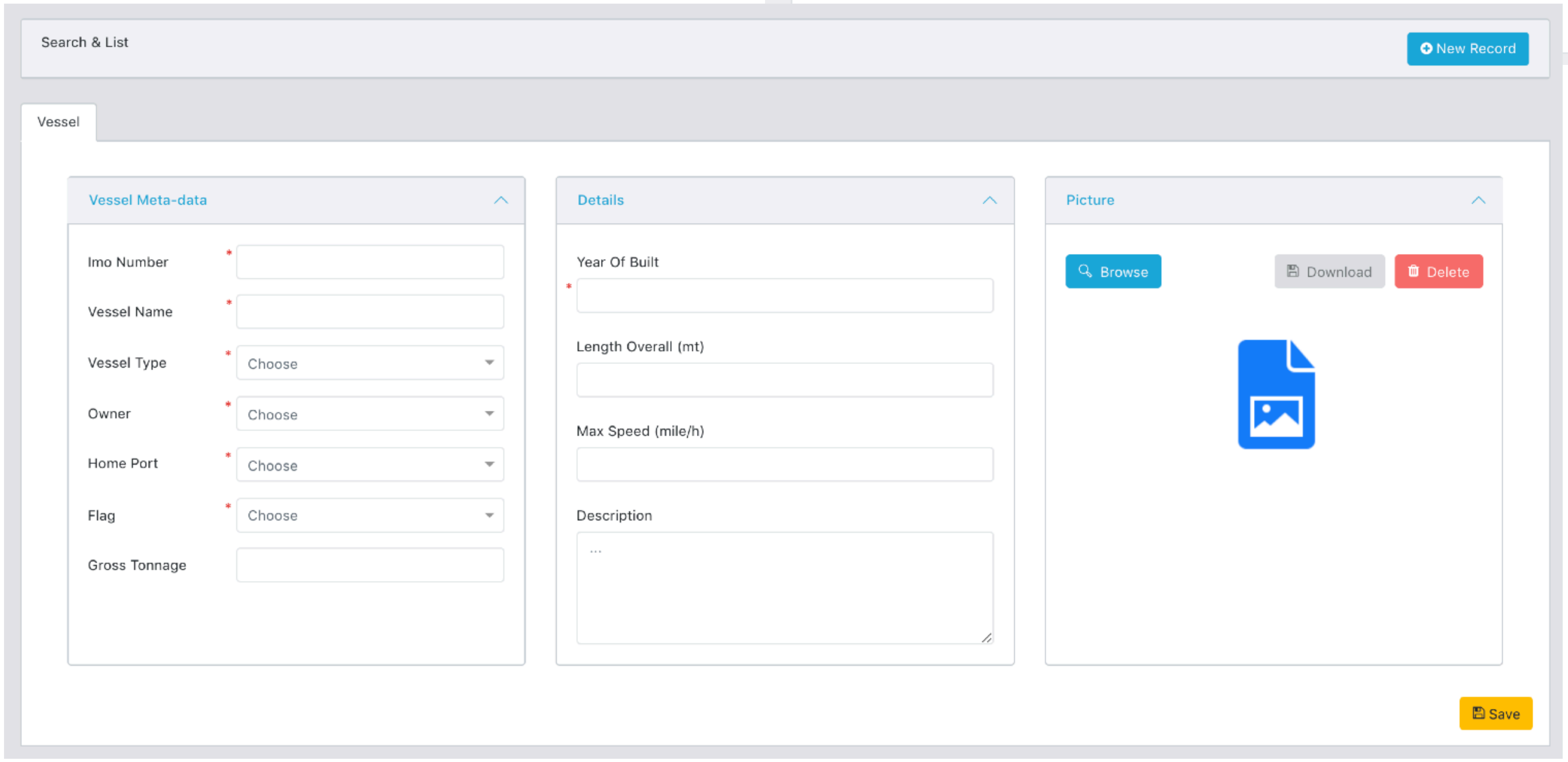
Dashboard / Vessel

Search & List

Vessel Title

Imo Number *		Vessel Name *	
Owner *	Choose	Home Port *	Choose
Flag *	Choose	Gross Tonnage	
Year Of Built		Length Overall	
Max Speed		Description	
Vessel Picture		Vessel Type *	Choose

After



Search & List

Vessel

Vessel Meta-data

Imo Number *	
Vessel Name *	
Vessel Type *	Choose
Owner *	Choose
Home Port *	Choose
Flag *	Choose
Gross Tonnage	

Details

Year Of Built *

Length Overall (mt)

Max Speed (mile/h)

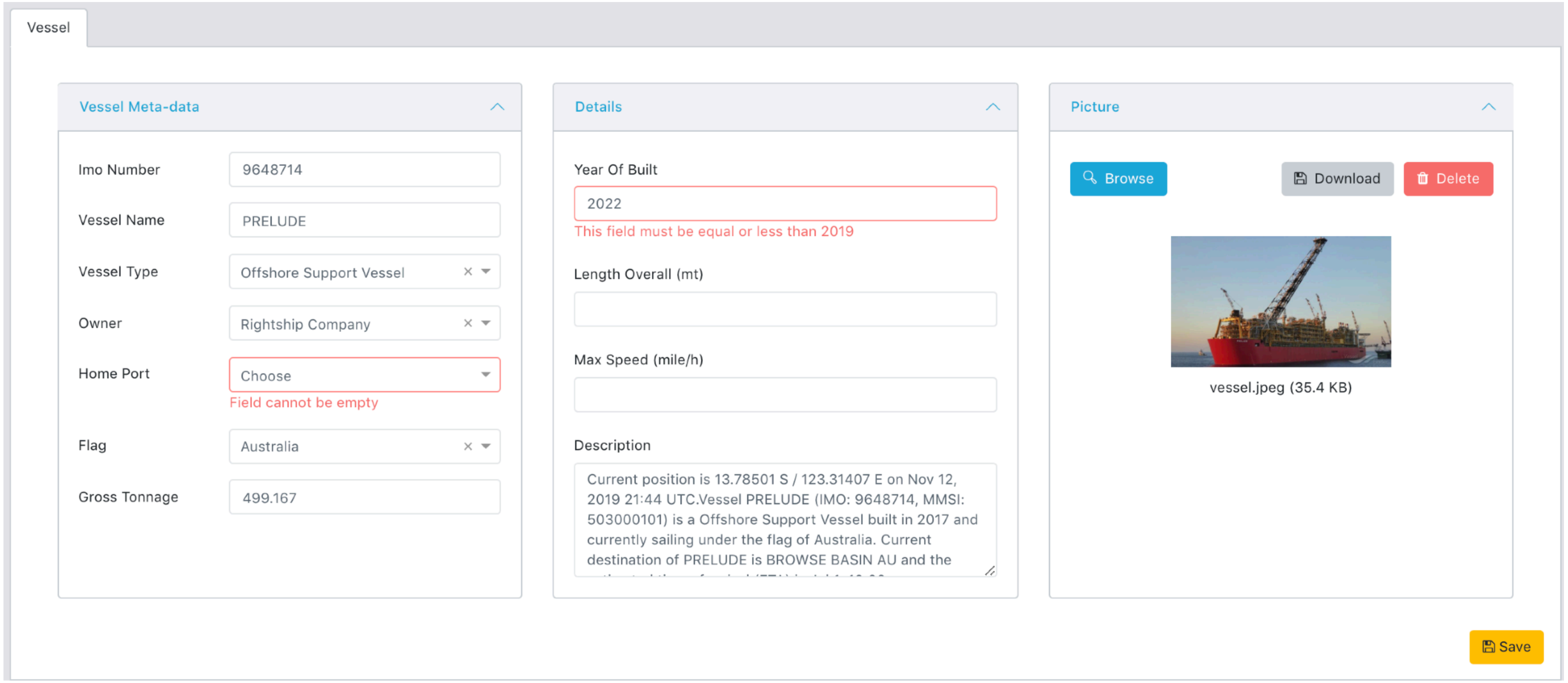
Description

...

Picture

NET CORE GENESIS

Frontend / Vessel page edited



Vessel

Vessel Meta-data

Imo Number	9648714
Vessel Name	PRELUDE
Vessel Type	Offshore Support Vessel
Owner	Rightship Company
Home Port	Choose Field cannot be empty
Flag	Australia
Gross Tonnage	499.167

Details

Year Of Built: 2022

This field must be equal or less than 2019

Length Overall (mt):

Max Speed (mile/h):

Description: Current position is 13.78501 S / 123.31407 E on Nov 12, 2019 21:44 UTC.Vessel PRELUDE (IMO: 9648714, MMSI: 503000101) is a Offshore Support Vessel built in 2017 and currently sailing under the flag of Australia. Current destination of PRELUDE is BROWSE BASIN AU and the

Picture

vessel.jpeg (35.4 KB)

NET CORE GENESIS

Congrats

You've

* Decreased your development effort by 70%

* Saved several months and ten thousands of dollars

Please refer to the web site and documentation
for details or contact us

www.NetCoreGenesis.com

For a live demo please visit and login with test@test.com and 123456

demo.NetCoreGenesis.com